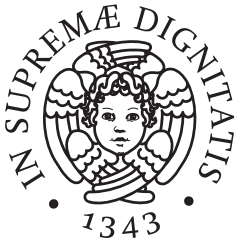
University of Pisa
Department of Civil and Industrial Engineering



Bachelor's Degree Thesis

Benchmark on the Aerodynamics of a 5:1 Cylinder using NEK5000

Supervisors:
Prof. Maria Vittoria Salvetti
Ing. Lorenzo Siconolfi
Ing. Alessandro Mariotti

Candidate:
Poojan Timilsina

Academic Year 2014/2015

Department of Civil and Industrial Engineering

APPROVED BY

Head of Department

……

(Signature)

……

(Name,Surname)

……..

(Date)

Poojan Timilsina

BENCHMARK ON THE AERODYNAMICS OF A 5:1 CYLINDER USING NEK5000

Bachelor's degree final work, Aerospace Engineering

Supervisor …….. …… ……

(Title, Name, Surname) (Signature) (Date)

Consultant …… …….. ……..

(Title, Name, Surname) (Signature) (Date)

Consultant …… …….. ……..

(Title, Name, Surname) (Signature) (Date)

Pisa, 2015

To my grandfather(BA)

Abstract

During this decade, the interest in the aerodynamics around a steady rectangular cylinder has grown up so vastly that various researchers and scientist working on bluff body aerodynamics contributed to the Benchmark on the Aerodynamics of a 5:1 cylinder (BARC) launched in 2008. These contributions are experimental or numerical using various simulation approaches like URANS, LES, VMS, hybrid URANS-LES, VMS-LES etc. This work is a numerical contribution based on Large-eddy simulations (LES) carried out using a spectral element open-source code,NEK5000.

The flow Reynolds number is kept high (40000) so that the presence of turbulence could be observed in the stationary sharp-edged rectangular cylinder. It is attempted to use the dynamic Smagorinsky model and the single filtering model to close the LES equations. One of the examples given by NEK producers called turbChannel is verified and used to have the desired code to run the simulation. Both 2D and 3D simulations are done and the obtained results like bulk parameters, mean value and RMS of the pressure coefficient along the cylinder surface and streamlines of the mean velocity field are tried to compare with the result obtained by *Ribeiro* for the 2D case and *Bruno et al.* for the 3D case.

Contents

List of Abbreviations

BARC	Benchmark on the Aerodynamics of a 5:1 Cylinder
DNS	Direct Numerical Simulation
LES	Large Eddy Simulation
NEK	NEK5000
RANS	Reynolds-Averaged Navier-Stokes
RMS	Root Mean Square
SEM	Spectral Element Method
SGS	Sub-grid scale modelling
VMS	Variational MultiScale
URANS	Unsteady Reynolds-Averaged Navier-Stokes
Wt	Weight

Nomenclature

Symbol	Description	Unit(s)v
δ	boundary layer thickness	$length$
Δ	Divergence	-
η	Kolmogorov length scale	$length^4$
ϵ	Average rate of dissipation of kinetic energy	$length^2/time^3$
ν	Kinematic viscosity	$length^2/time$
ρ	Density	$mass/length^3$
τ	Kolmogorov time scale	$time^2$
τ_{kk}	subgrid-scale stresses	$time^2$
C	Courant Number	$Dimensionless$
C_D	Drag Coefficient	$Dimensionless$
C_L	Lift Coefficient	$Dimensionless$
C_P	Pressure Coefficient	$Dimensionless$
I_x	Free stream turbulence	$Dimensionless$
L	Upper limit of integral scale	$Dimensionless$
L_{ij}	Resolved stress tensor	-
N	Number of points along a given mesh direction	$Dimensionless$
P	Pressure	Pa
P_∞	Upstream Pressure	Pa
Re	Reynolds Number	$Dimensionless$
Re_τ	Friction Reynolds Number	$Dimensionless$
S_{ij}	Rate-of-strain tensor	-
St_D	Strouhal number	$Dimensionless$
T_{ij}	Stress tensor	-
$Vrms$	RMS velocity in y direction	$length/time$

1 Introduction

1.1 Generalities

The Benchmark on the Aerodynamics of a Rectangular 5:1 Cylinder (BARC) aims at establishing a platform for discussion among scientists working on bluff body aerodynamics, and in particular it deals with the analysis of the turbulent, separated flow around an elongated rectangular cylinder. In July 2008 BARC was announced during the VI Colloquium on Bluff Body Aerodynamics and Applications (BBAA VI). From there onwards, the researchers from all around the world started working on BARC and in 2014 the results of about 70 realizations of the BARC flow configuration obtained under a nominally common set-up in both wind tunnel experiments and numerical simulations(LES, URANS and hybrid URANS/LES) were compared among themselves. [1] Near the wake flow, the base pressure and the drag coefficient obtained in the different flow realizations are in very good agreement. On the other hand significant differences were observes in the behavior of the flow and of the aerodynamics loads on the cylinder lateral sides.

The characteristics of the flow field around rectangular bodies is of great interest both for fundamental research and for applications. From the fundamental research point of view, in spite of the simple and nominally two-dimensional geometry, the flow over an elongated rectangular cylinder at high Reynolds numbers is highly complex, being three dimensional, turbulent and characterized by unsteady flow separation and reattachment.
On the other hand, thanks to the simple geometry, a detailed analysis of the flow dynamics can be carried out. Also different patterns can be identified also occurring when dealing with more complex geometries. As for applications, this benchmark problem provides useful information on the aerodynamics of a wide range of bluff bodies of interest in civil engineering (e.g. long-span bridge decks or high-rise buildings) as well as in other engineering areas.[1]

1.2 Aims and Requirements

BARC addresses the high Reynolds number, external, unsteady flow around a stationary, sharp-edged rectangular cylinder, and the associated aerodynamic actions. The breadth(B) to depth(D) ratio is set equal to 5 (it is this Benchmark initial condition set by the researchers) because it is characterized by shear-layers detaching at the upstream cylinder corners and reattaching on the cylinder side rather close the downstream corners.[3] As BARC assesses the consistency of wind tunnel measurements and computational results, it develops integrated procedures relying on both experimental and computational outcomes. This research consists

of LES simulations of BARC by using the open-source code, NEK5000, so the requirements would be focused on the numerical part.[2]

1.3 BARC configuration for this research

The depth-based Reynolds number $Re_D = UD/\eta$ has to be in the range of $2 * 10^4$ to $6 * 10^4$. The angle of incidence $\alpha = 0$, i.e. the incoming flow has to be set parallel to the breadth of the rectangle. Besides, the maximum intensity of the longitudinal component of the freestream turbulence (I_x) has to be lower than 0.01 and the spanwise length of the obstacle (L/D) has to be greater than or equal to 3.

In this analysis, Re_D is set to $4 * 10^4$ and I_x is set to 0.01. The L/D ratio is set to 5. The geometry of the spatial domain is shown in Figure 1.

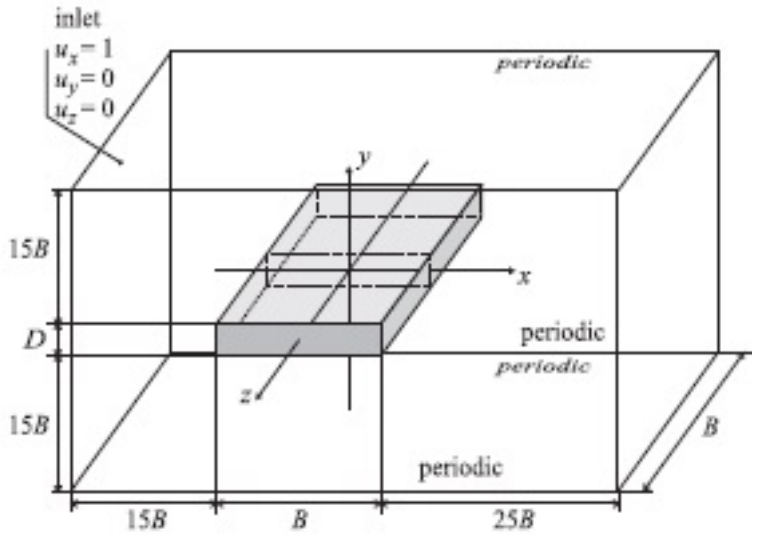


Figure 1: Computational studies:model and domain geometry

In this research, Large Eddy Simulation(LES) will be used with NEK5000 as the programming tool. Hereafter, this simulation method is discussed briefly and is also compared to Direct Numerical Simulation(DNS) method.

2 Large Eddy Simulation (LES)

LES is a mathematical model for turbulence used in computational fluid dynamics. The principle operation in LES is a low-pass filtering that is applied to Navier-Stokes equations to eliminate small scales of the solution in order to reduce

the computational cost of the simulation. By doing so, the solution obtained is a filtered velocity field.
Thus, LES resolves large scales of the flow field solution allowing better fidelity comparing to the RANS(Reynolds Averaged Navier-Stokes) and also models the smallest scales of the solution unlike Direct Numerical Simulation (DNS). [5]

DNS is one of the methods of solving Navier-Stokes equation numerically without any turbulence model. Here, the whole range of spatial and temporal scales of turbulence must be resolved. The spatial scales are resolved up to the smallest dissipative scales called the kolmograv microscales (η).
It is known that

$$\eta = (\nu^3/\epsilon)^{1/4}$$

where ν is the kinematic viscosity and ϵ is the rate of dissipation of kinetic energy. For temporal scales, $N * h > L$ where, N is the number of points along a given mesh direction, h is its increment and L is the upper limit of the integral scale. In most cases it is estimated that $h \leq \eta$. Knowing that $\epsilon = (V_{rms}^3)/L$ and Reynolds number$(Re) = V_{rms}^3 * L/\eta$, it can be concluded with the following expression:

$$N^3 \geq Re^{2.25}$$

Observe that the memory requirement grows very fast with the Reynolds number. Taking into note that the turbulence time scale $\tau = L/Vrms$, as a final result, it is obtained that $\tau \sim Re^3$. Thus,DNS is not affordable at high Reynolds numbers typical of practical applications. [4]

A small look over the properties of the filters, and their definitions is done.
It is important to understand the term Sub-grid Scale Modelling (SGS) that refers to the representation of important small-scale physical processes that occur at length-scales that cannot be adequately resolved in a computational mesh. In LES, SGS modelling is used to represent the effects of unresolved small-scale fluid motions ($smalleddies, vortexes$) in the equations governing the large-scale motions that are resolved in computer models.[7]

2.1 Definition of Filters

1. $\bar{f}(X,t) = \int f(X',t)\, G(X'-X)\, dX'$ where $G(X'-X)$ is a normalized filter function and $\int G(X'-X)\, dX' = 1$

2. In 3D the filter function is:$G(X'-X) = \sum_{\alpha=1}^{3} G\alpha\,(x'\alpha - x\alpha)$ and x_α is the filter direction.

2.2 Properties of Filters

1. Mathematical Properties
   Assuming $u = u(\vec{x}, t), v = v(\vec{x}, t)$ and c is a constant.
   (a) $\overline{cu} = c\overline{u}$ and $\overline{c} = c$
   (b) $\overline{u + v} = \overline{u} + \overline{v}$
   (c) $\frac{\overline{\delta u}}{\delta x_i} = \frac{\delta \overline{u}}{\delta x_i}$
   (d) For many filters: $\overline{u} \neq \overline{u}, \overline{u}' \neq 0$ where $u' = u - \overline{u}$
2. Spatial Properties
   Considering that the coefficient of Fourier of u ($\hat{u}(k)$) and of the filtered variable ($\hat{\bar{u}}(k)$), by the means of convolution theorem it is obtained that: $(\hat{\bar{u}}(k)) = \hat{G}(k)\hat{u}(k)$ where $\hat{G}(k)$ is the Filter function G in the space of wave numbers.

2.3 LES formulation

1. Incompressible fluid
   (a) $\frac{\delta u_i}{\delta x_i} = 0$
   (b) $\rho \frac{\delta u_i}{\delta t} + \rho u_j \frac{\delta u_i}{\delta x_j} = -\frac{\delta p}{\delta x_i} + \mu \Delta u_i$

   After applying a filter in the Navier-Stokes equation and using the commutative property:
2. The continuity equation is reduced to: $\frac{\delta \bar{u}}{\delta x_i} = 0$
3. The momentum equation is reduced to: $\frac{\delta \overline{u_i}}{\delta t} + \frac{\delta \bar{u}_i \bar{u}_j}{\delta x_j} = -\frac{1}{\rho}\frac{\delta \bar{p}}{dx_i} + 2\upsilon \frac{\delta \bar{S_{ij}}}{\delta x_j} - \frac{\delta \tau_{ij}}{\delta x_j}$ where $\tau_{ij} = \overline{u_i u_j} - \bar{u}_i \bar{u}_j$ is the subgrid stress tensor and this term must be modeled to close the LES equations.

2.4 Methods for Modelling Subgrid

2.4.1 Smagorinsky Model(1963)

The sub-grid tensor is modeled with the addition of sub-grid viscosity obtaining the following equation:

$$\tau_{ij} - \frac{\delta_{ij}}{3}\tau_{kk} = -2C\bar{\Delta}^2 |\bar{S}| \bar{S_{ij}} \text{ where } |\bar{S}| = (\bar{S}_{ij}\bar{S_{ij}})^{\frac{1}{2}}$$

Therefore it is concluded that τ_{ij} has its own viscous term whose value is $v_s = 2C\bar{\Delta}^2|\bar{S}|$ and it can be considered as a sub-grid viscosity (Eddy-viscosity). [8]

But unfortunately, this model presents some drawbacks like:

1. The constant C must be assigned a priori.
2. Predicts a wrong asymptotic behavior near the walls or in a laminar flow.
3. Does not permit the passage of energy of small scales to large scales.
4. It assumes that the main axes of τ_{ij} are parallel to those of the tensor of velocity of deformation $\bar{S}_{ij}$.

2.4.2 Eddy-Viscosity Dynamic Model(Germano et al. 1991)

The basic idea of this model is to utilize the algebraic expression demonstrated by Germano and calculate the constant (C) of the Smagorinsky model using the smallest scale resolved.
The mathematical formulation is expressed below:

1. The Smagorinsky model is utilized as the basic model:

$$\boxed{\tau_{ij} - \frac{\delta_{ij}}{3}\tau_{kk} = -2C\bar{\Delta}^2|\bar{S}|\bar{S}_{ij}}$$

2. Another filter (test filter) of higher dimension than that associated to the grid is applied. The stress tensor of subtest is:

$$\boxed{T_{ij} = \widehat{\overline{u_i u_j}} - \hat{\bar{u}}_i \hat{\bar{u}}_j}$$

3. The Smagorinsky model is also utilized for the subtest tensor:

$$\boxed{T_{ij} - \frac{\delta_{ij}}{3}T_{kk} = -2C\hat{\Delta}^2|\hat{\bar{S}}|\hat{\bar{S}}_{ij}}$$

4. By Germano it is demonstrated that:

$$\boxed{L_{ij} = T_{ij} - \hat{\tau}_{ij} = \widehat{\bar{u}_i \bar{u}_j} - \hat{\bar{u}}_i \hat{\bar{u}}_j}$$

5. By using the expressions at the sub-grid and subtest tensor given previously to the **Smagorinsky model**, it is obtained that:

$$L_{ij} = -2C\overline{\Delta}^2 M_{ij}$$

where $M_{ij} = \frac{\hat{\Delta}^2}{\overline{\Delta}^2}|\hat{\bar{S}}|\hat{\bar{S}}_{ij} - \widehat{|\bar{S}|\bar{S}_{ij}}$

6. Finally C can be as follows:

$$\boxed{C = -\frac{L_{ij}M_{ij}}{2\overline{\Delta}^2 M_{ij}M_{ij}}}$$

This model is advantageous compared to the Smagorinsky one as there is no necessity of assigning the value of C a priori because it can be calculated for each point of the grid and for each time step. Moreover, this model has the correct asymptotic behavior (C=0) near the wall and also for the laminar fluids. Besides, it permits the flow of energy from small scales to the resolved ones and it is easily generalized to compressible flows.

But the problem of this model is the coefficient C presents strong non physical fluctuations. Moreover, the Smagorinsky model is used as the base model that means the axes of the stress tensor of subgrid are aligned with those of the strain rate tensor S_{ij}. The assumptions behind this are not verified in most flows.

In order to avoid the oscillation problem of the coefficient C while it is calculated dynamically, an operation of averaging in the directions in which the flow is homogeneous can be done. This procedure could be done only for cases having homogeneous directions.[9]. It is worth to remind that the BARC case that is going to be analyzed is homogenous in z-direction.

3 Introduction to NEK5000

NEK5000 is an open-source computational fluid dynamics solver based on the spectral element method. The code is written in Fortran77 or C[6]. It is based on the Nekton 2.0 spectral element code written by Paul Fischer, Lee Ho, and Einar Ronquist in 1986-1991, with technical input from A. Patera and Y. Maday. The Nek5000 development was continued from the 90s through present by P. Fischer, Jerry Kruse, Julie Mullen, Henry Tufo, James Lottes, Chaman Verma, and Stefan Kerkemeier. [10]
NEK5000 is based on the Spectral Element Method(SEM), which is briefly described in the following.

3.1 Spectral Element Method

The Spectral Element Method is a highly accurate numerical method that combines the flexibility finite-elements with the accuracy of a spectral method.

Spectral methods involve the expansion of the solution to a differential equation, in a high-order orthogonal polynomial expansion, the coefficients of which are determined by a weighted-residual projection technique. The schemes could be infinite order accurate but all depends on how properly is the expansion polynomials chosen.

The finite-element procedure is a weighted-residual technique applied to a series of expansions, each with support over an element (a small region of space). When the mentioned technique is directly derived from an associated variational principle, continuity of natural boundary conditions is implicitly satisfied at element boundaries.

In some cases finite-element and spectral methods could be exactly same but the main feature of spectral techniques is accuracy but general, complex flow problems are typically extremely difficult to be solved using spectral methods. On the other hand, the main advantage of finite-element method is generality; elements are typically chosen to be at most quadratic $[2-4]$, and consequently, great accuracy is only achieved with difficulty. [11]

3.2 Introduction to NEK files

First of all, it is important to learn the basic utilities of NEK5000 and in order to do so, downloading the NEK5000 examples provided by the NEK developers is the best way. To have a basic idea, each simulation in NEK5000 is defined by three files, the .rea file, the .usr file and the SIZE file. There are other files like the .box file which is used to generate the .rea file by defining the mesh and the boundary condition, the .map file that is generated from the .rea file by running genmap.

To run a case the key steps necessary are :

1. Setup the case specific files: .usr file, .rea file and SIZE file.
2. Create a .box file containing the parameters needed to build the mesh.
3. Generate the mesh by using genbox.
4. Run the domain partitioning tool genmap.
5. Build NEK5000, to do so, copy makenek from $nek5_svn/trunk/nek$
   to the working directory and compile the code.(./makenek [name of simulation])
6. Run the code.(nek [name of simulation])

7. Finally analyze the checkpoint data stored in the field files (e.g. name.f00001) using a visualization tool (e.g. Visit, paraview etc.) [12]

The scheme of the desired procedure is represented in Figure 2.

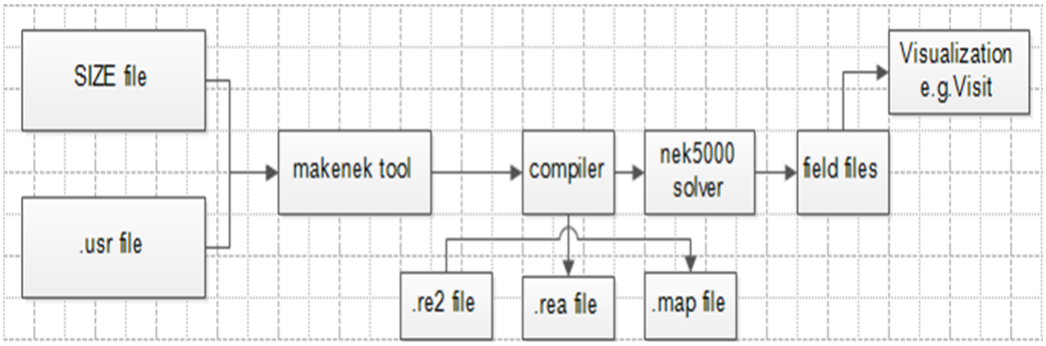


Figure 2: Scheme for NEK simulation

Among the examples provided by NEK developers, turbChannel is useful since it concerns LES of a Turbulent channel flow.
BARC consists of passing a turbulent flow through a small cylinder. Using the codes of turbChannel and changing some parts of it would be a good way to do the BARC but before that, turbChannel must be verified.
BARC will be implemented by changing some of the codes of turbChannel. Two SGS models i.e. simple filtering model and dynamic Smagorinsky model are available. The idea of simple filtering model is to use a simple low-pass filter without any explicit SGS model. The filter functions could be defined in the .rea file by defining the filter weight in parameter 103 and by specifying the additional modes to filter in parameter 101 (see turbChannel.rea file).

Thus test-case provides a first validation of these two models which could be eventually used.

3.3 LES of the Turbulent Channel flow

For the verification of simple filtering method, various simulations for different number of modes and different weights are carried out. The total number of elements of this grid is 512 and its polynomial order is 8. The Reynolds number used is 10935 and the friction Reynolds number($Re\_tau = 590$) Mean velocity and rms fluctuation profiles in wall units are represented and the data obtained by A. Abb(Politecnico di Milano) and M. Germano (Politecnico de Torino) during their LES of fully developed turbulent channel flow at $Re\_tau = 590$ are conserved for comparison.

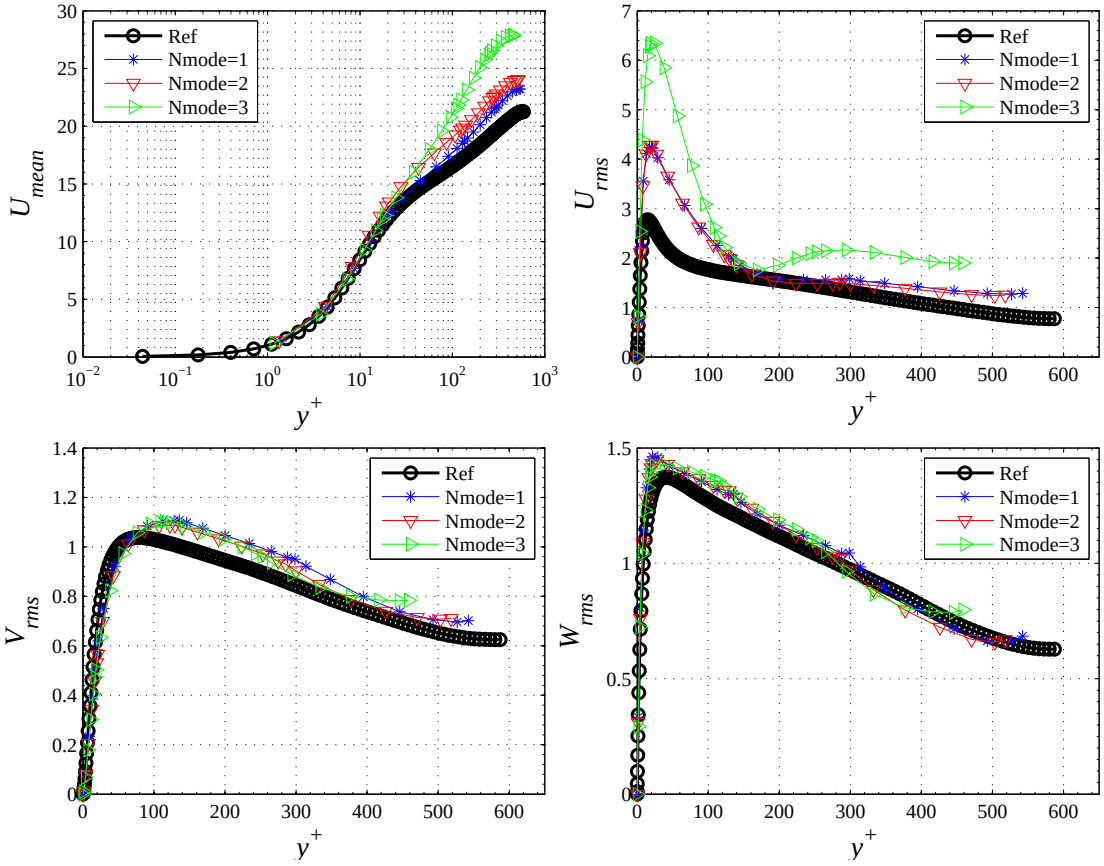


Figure 3: The mean and rms velocity profile (W-filt=0.01)

Figures 3, 4 and 5 shows that the weight of the filter have a small influence in the final result and the most important factor is the number of modes. When the number of filtered modes is 1 and 2 the result obtained is more similar to the reference data. Besides, the rms velocity in x-direction varies the most for y^+= 100 in all three cases.

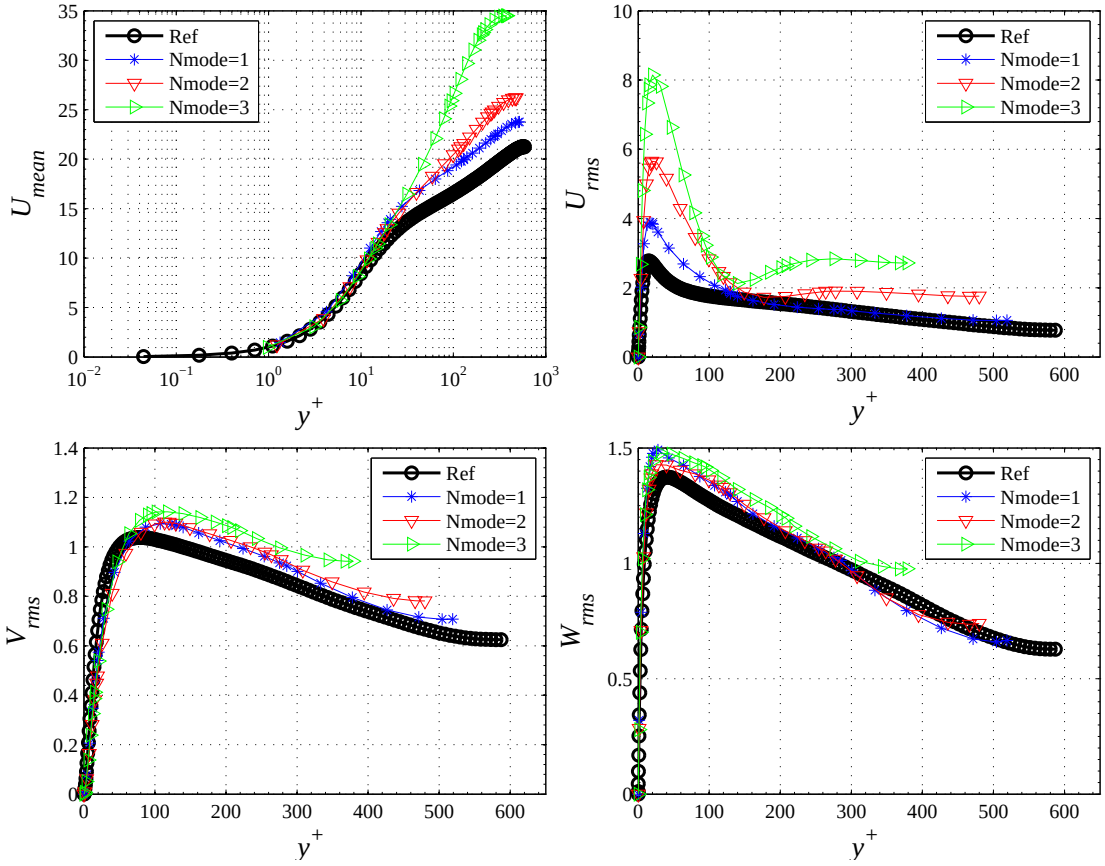


Figure 4: The mean and rms velocity profile (W-filt=0.025)

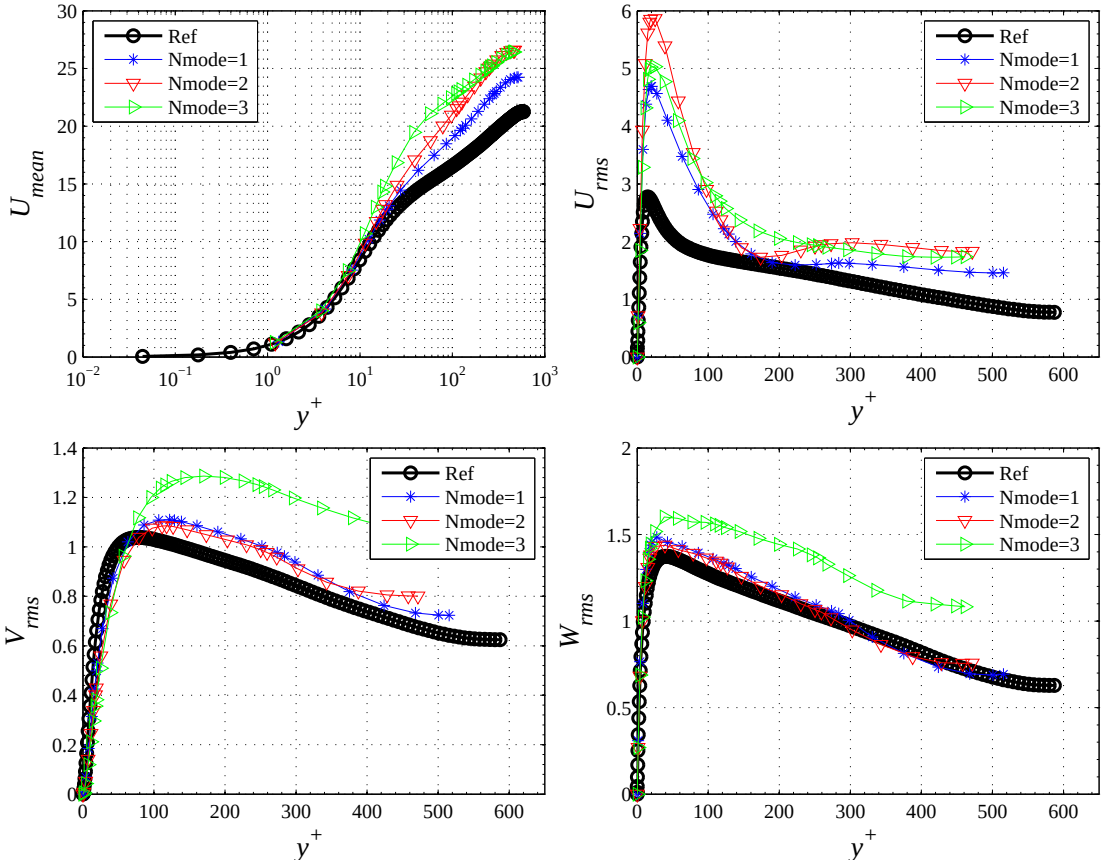


Figure 5: The mean and rms velocity profile (W-filt=0.05)

Even though, there is a slight variation on the final result, this test validates the turbChannel single filtering model as the results obtained are similar to the reference result. Now the BARC simulation could be started for the single filter model.

3.4 BARC on NEK5000

After verifying the turbChannel example, it is possible to use it for the BARC implementation. As stated earlier, turbChannel is a turbulent flow model which is homogenous in y and z direction. BARC consists of homogenous flow only on z direction. This implies that the code in the .usr file has to be slightly modified but before, a new mesh should be created modifying the box file.
The mesh of turbChannel is a single box but BARC consists of a cylinder within the box. The best and the simplest way is to create various boxes with a gap between the boxes and put the boundary condition as a wall so that it acts like

a cylinder. In order to build a mesh for a 3D simulation there are two ways, by directly defining a 3D mesh or by creating a 2D mesh and later extruding it in the z-direction. Creating a 3D mesh directly would be easier but during the analysis it is found that the result obtained is good if only one processor is used but if it is not so then there might have a problem while distributing the number of elements per each processor as the surfaces are not distributed homogeneously as seen in Figure 6. But if it is first done in 2D and later extruded than the surfaces are distributed homogeneously and there won't be any problem during the elements distribution for each processor.

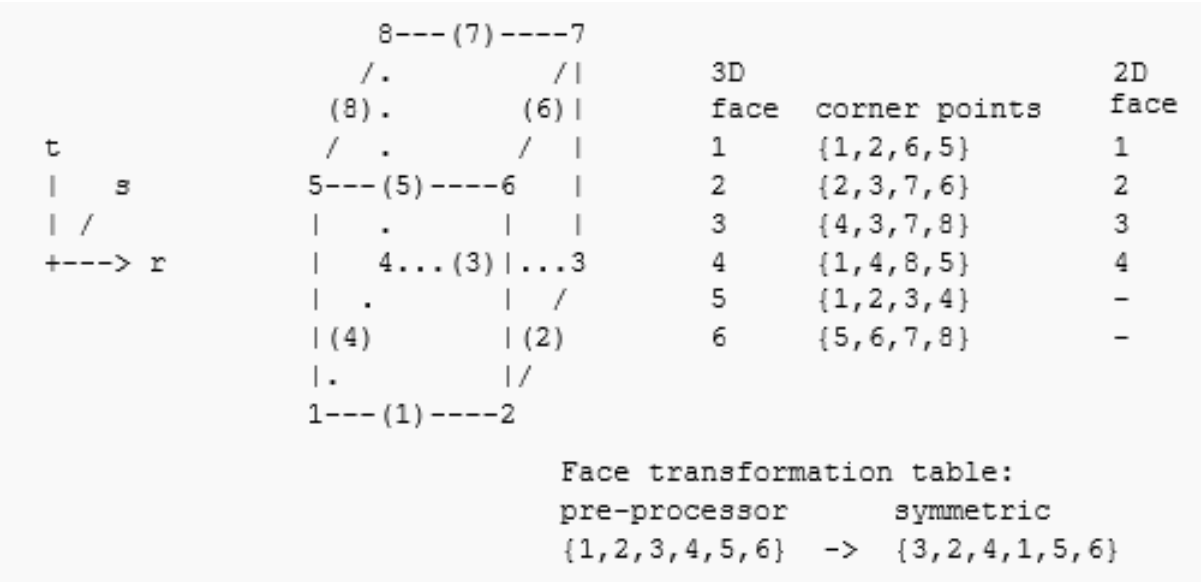


Figure 6: Representation of element boundaries

For that reason a 2D .box file is created (whose code could be found in Appendix A) and later it is extruded by using the command $n2to3$. Finally the mesh obtained is shown in Figure 7. In Figure 8 it could be observed that the number of elements around the cylinder are tried to be distributed homogeneously.

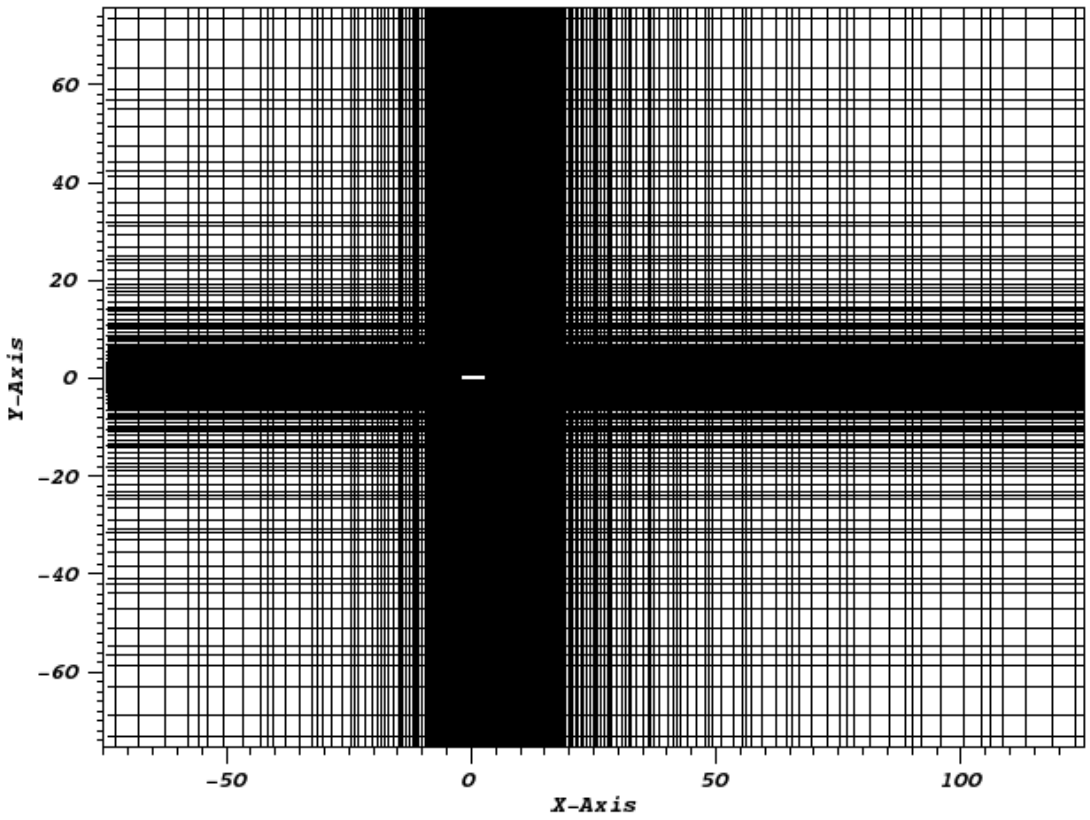


Figure 7: Mesh in NEK5000

The mesh most often used to do the BARC simulation is as presented in Figure 9 which is more refined around the surrounding of the cylinder and more coarse away from the cylinder. Due to the restriction of NEK5000 (number of elements on the boxes lying on same axes must be the same), a mesh like in Figure 7 is obtained.(More detail is presented in Appendix A).

After the creation of the mesh, the modification of the code is started. In NEK5000 the .usr file is the main platform that contains the necessary code to be run.

The turbChannel .usr file contains the code necessary to obtain a turbulent flow around the cylinder but some changes should be made in order to have the z-direction homogeneous.A subroutine called linear_average_s is created that calculates the overall average of the desired parameter. TurbChannel has a similar type of subroutine called planar_average_s that calculates the planar average. The same procedure as in the planar average is followed and only the necessary code is changed in order to obtain the average. (More detail is presented in Appendix D) After doing the linear average subroutine, another subroutine called my_avg_all

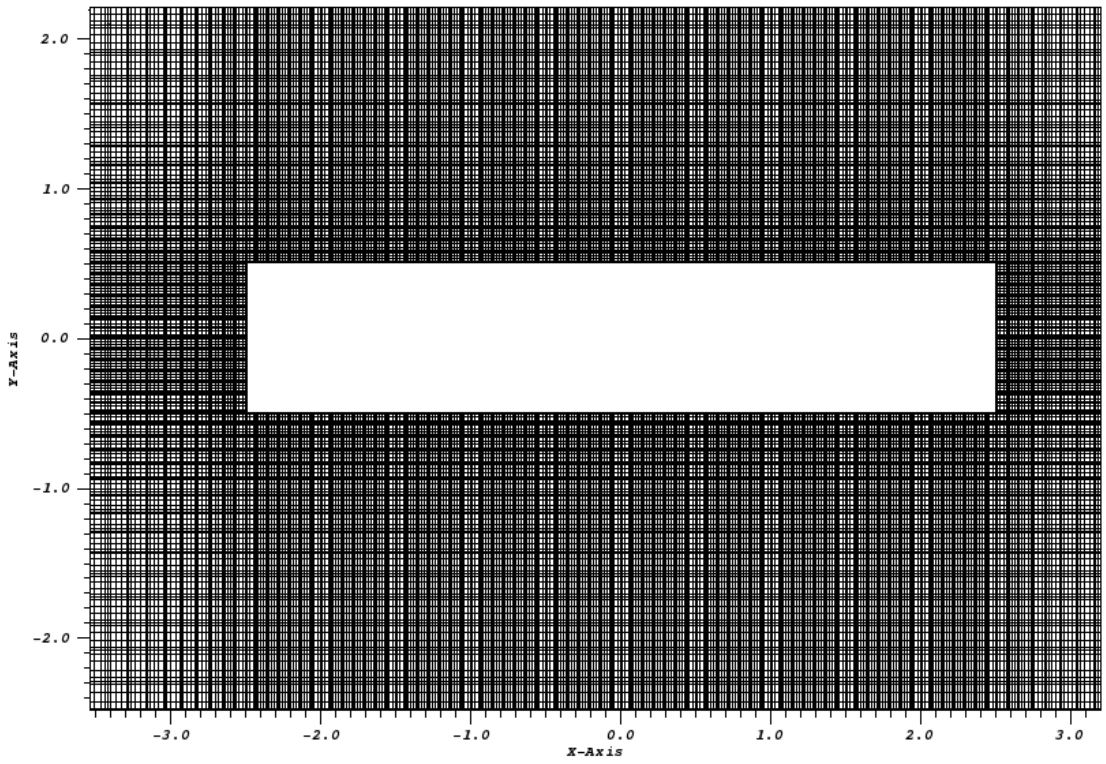


Figure 8: Zoom in around the cylinder

that does the postprocessing and computes the average in z-direction is created(More detail of the subroutine is presented in Appendix D).After finishing the coding procedure, the number of time steps, the Reynolds number, and all other important data should be fixed on the .rea file. (The code of the .rea file is presented in Appendix B)

3.5 Preliminary verification-2D

Instead of starting to simulate the 3D case directly, it is faster and cheaper to first verify the code in 2D by using a small mesh and compare the obtained result with the reference results. Once verified then the 3D simulation could be started. (The code of the smaller box file is presented in Appendix A).This smaller grid has 1090 elements and the order of polynomials is 6.

During the turbChannel verification, it was seen that the lesser the number of filtered modes, the closer are the results to the reference ones. So, for the code verification, the number of modes is fixed to 1 and the weights are varied to find out the best case. 15 different tests are done changing the weight from 0.01 till 0.5 and the obtained results can be observed in Table 1. Strouhal number ($St_D = \frac{f_s D}{U}$)

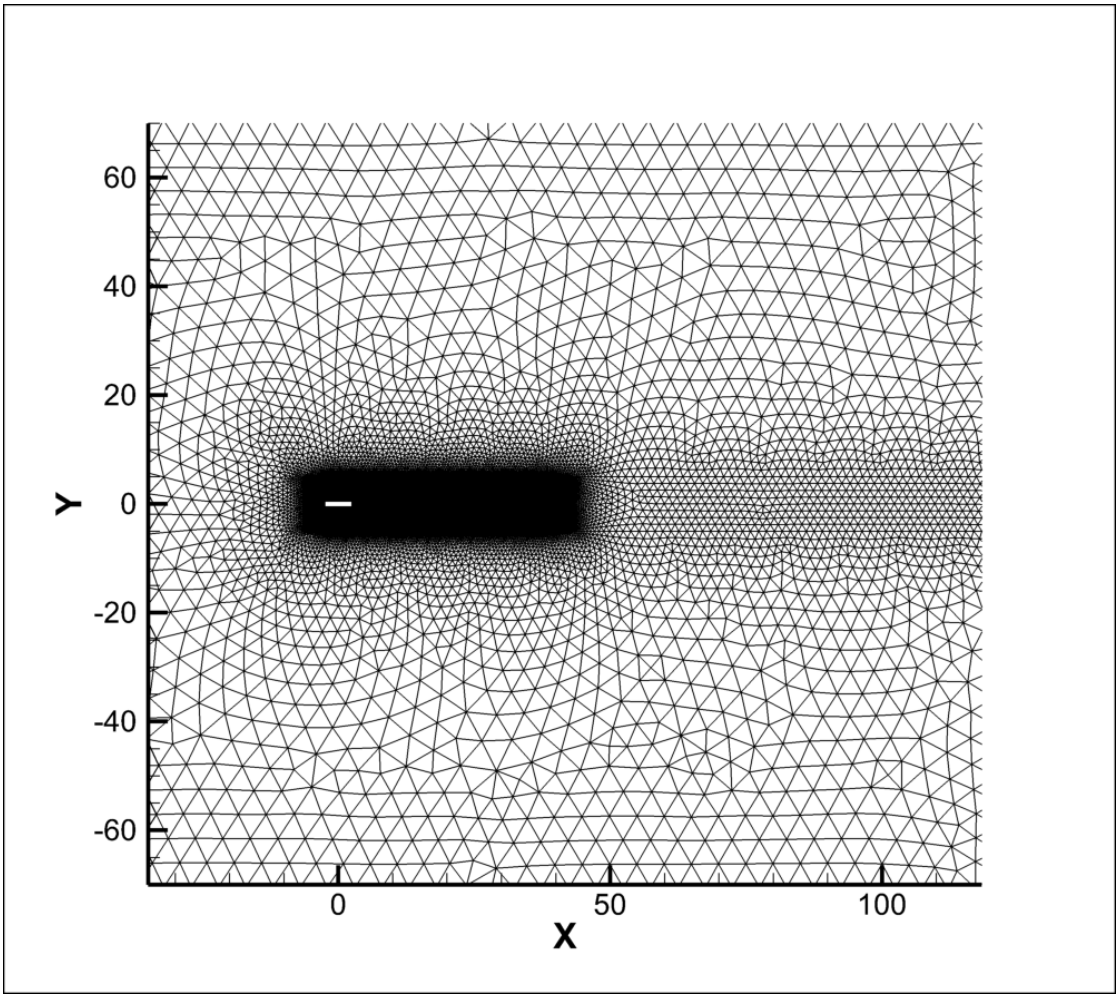


Figure 9: Commonly used mesh

for this analysis is equal to the shedding frequency (f_s) as D and U are 1. The shedding frequency is evaluated from the time fluctuations of the lift coefficient or from pressure or velocity time signals and in this case it is evaluated from the time fluctuations of the lift coefficient. [1]
As a reference data, results obtained by Ribeiro in 2011 for 40000 Reynolds number and 2D simulation despite of being simulated by using RANS model is taken.[1]. In the figure10, the velocity magnitude after 50000 iterations for weight 0.01 can be observed. As expected, vortex shedding is observed.

The Strouhal number in most of the simulation is higher (0.11- 0.12) than the obtained by Ribeiro. But also the strouhal number obtained for all most all these

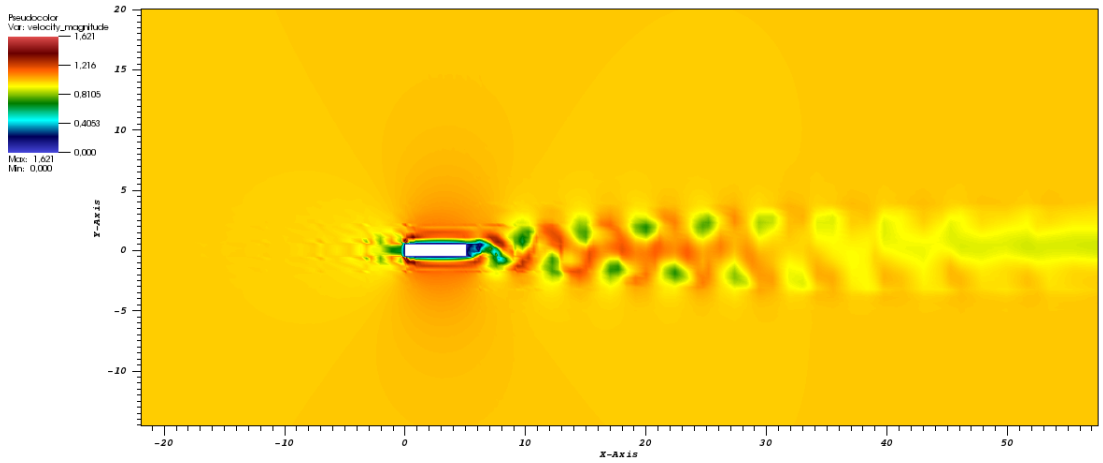


Figure 10: Velocity Magnitude after 5000 iterations: Weight:0.01

$Source$	t-avg(Cd)	t-avg(Cl)	t-std(Cl)	St_D
$Ribeiro(2011)$	1.170	-	0.900	0.073
$Arslan(2011)$	0.9849-1.39	-	0.59-0.84	0.107-0.16
$Weight$	$TestResult$			
0.01	1.1803	-0.0179	0.4915	0.192
0.02	1.1162	-0.0106	0.422	0.185
0.03	1.1398	-0.0154	0.486	0.181
0.04	1.1069	-0.0055	0.4714	0.176
0.05	1.0714	0.0063	0.4421	0.170
0.06	1.0359	$4.5217e10^4$	0.4036	0.169
0.07	1.0007	0.0097	0.3512	0.167
0.08	0.9656	0.0069	0.2822	0.169
0.09	0.9367	0.0103	0.2238	0.169
0.10	0.9177	0.0149	0.1936	0.169
0.20	0.8834	0.0196	0.1738	0.166
0.30	0.9286	0.0291	0.1833	0.158
0.40	0.9996	0.0314	0.2023	0.151
0.50	1.0947	0.0335	0.3224	0.142

Table 1: Bulk parameters: numerical results

cases are higher and the main reason behind this is because of the smaller mesh designed to do this validation and fewer number of elements around the cylinder. Moreover, for lower weights, the time-averaged C_D and C_L are almost similar to

reference result compared with the higher weights , in exception of Wt=0.5.
The drag and lift coefficients after convergence (after 1800 time steps) are represented in the Figures 11, 12, 13, 14.

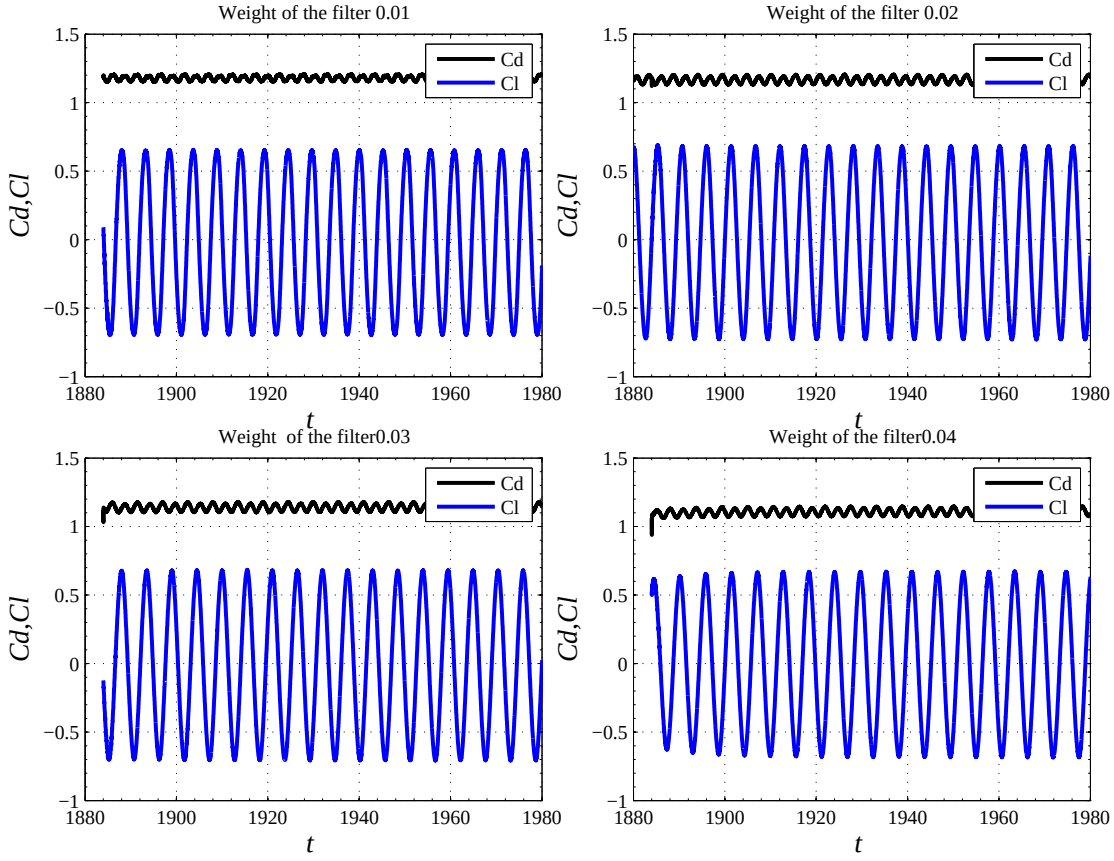


Figure 11: Lift and Drag Coefficients (Wt 0.01-0.04)

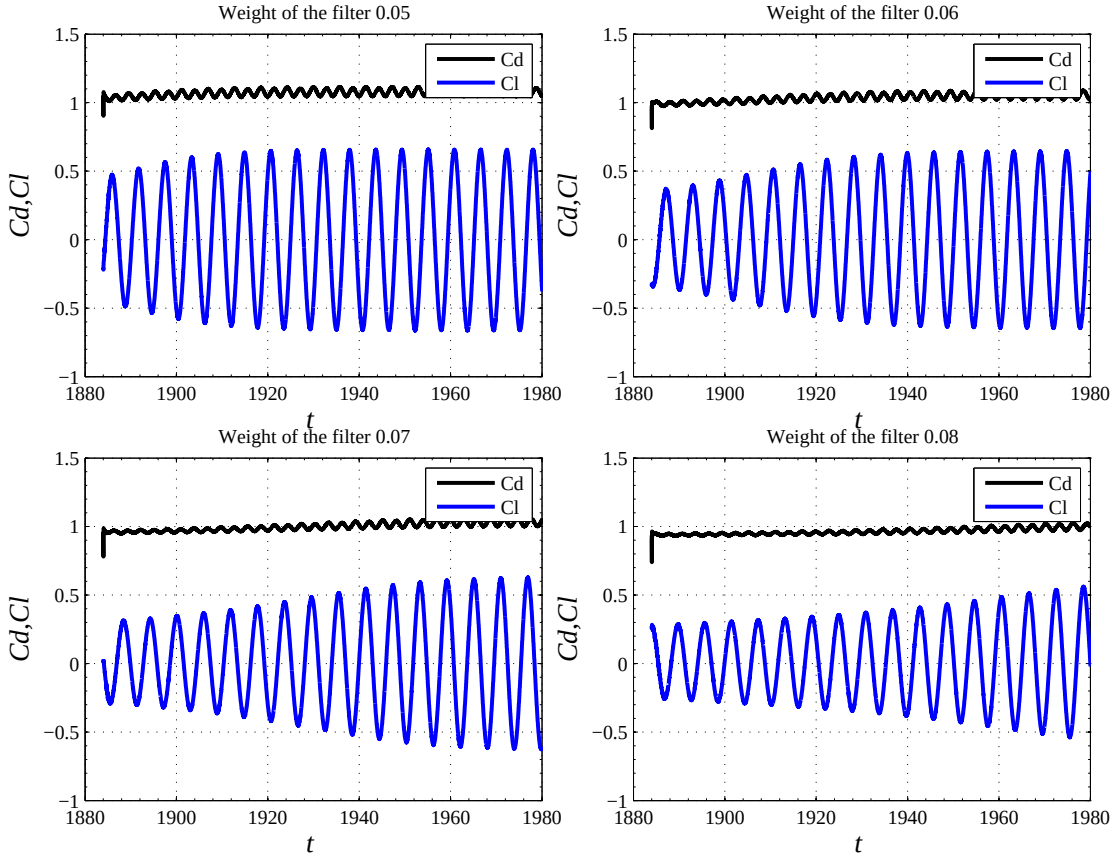


Figure 12: Lift and Drag Coefficients(Wt 0.05-0.08)

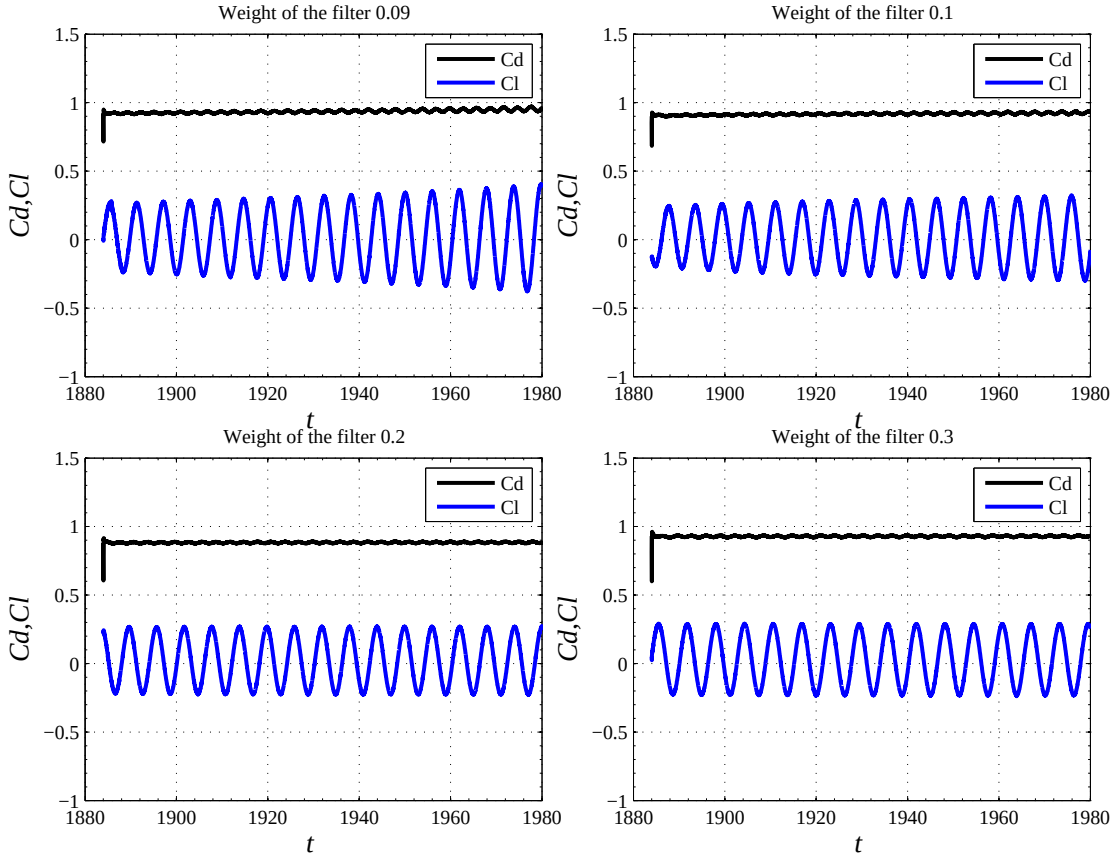


Figure 13: Lift and Drag Coefficients(Wt 0.09-0.3)

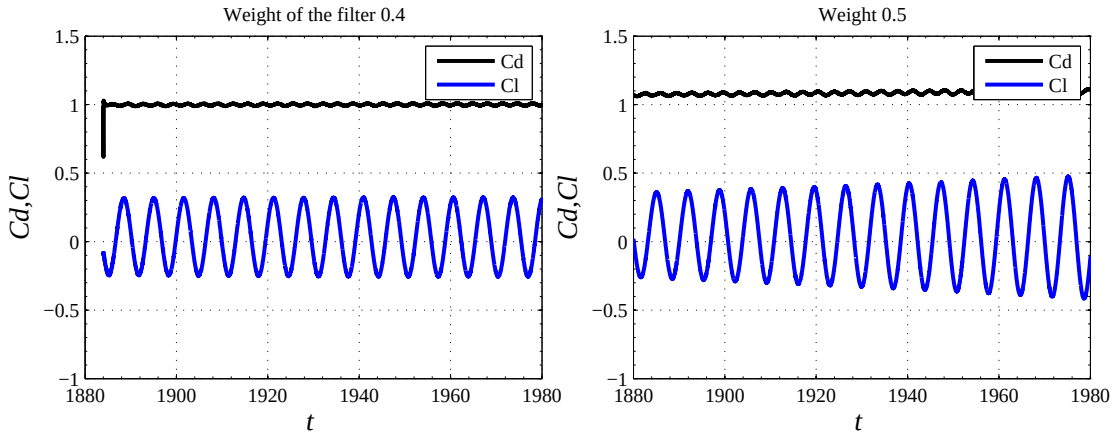


Figure 14: Lift and Drag Coefficients(Wt 0.4 and 0.5)

Observing that the test values are similar to the Reference value, in exception of St, the 3D simulation could be started but in this case the refined mesh would be used.

4 Final Results

4.1 Procedure

During the preliminary validation it is seen that for 1 and 2 number of modes the result is almost similar to the reference result. As for the 2D validation 1 number of modes is used so now, for the final simulation the number of modes of the filter is set to 2 and the weights to 0.05, 0.1 and 0.5.
As explained previously, to do the 3D simulation, the 2D mesh is created and it is extruded to 3D. This mesh consists of around 55 thousand number of elements in the whole field which is very high considering the 1090 number of elements of the small mesh that was used for the 2D during the code validation. Simulating such a higher number of elements requires a large number of processors and a lot of time so, in order to save the time, first of all a general simulation is done till obtaining total convergence (after almost 600 time steps). After that, the desired weights are varied and the simulations are restarted.

4.2 Results obtained

4.2.1 Bulk Parameters

The lift and the drag coefficients after obtaining the convergence are in the Figures 17, 15 and 16 for 0.5, 0.1 and 0.05 respectively and Figure 18 contains the Cd and Cl obtained by Bruno in 2010. [15] The bulk parameters are reported in Table 2 where $t-avg(C_D)$ and $t-avg(C_L)$ are the time- and spanwise-averaged drag and lift coefficients; $t-std(Cl)$ is the standard deviation of the time variation of the lift coefficient; St_D is equal to the shedding frequency (explained in subsection 3.5). Table 3 contains the bulk parameters obtained by different researchers on 3D using various numerical approaches till now. The results obtained by NEK5000 are similar to that of obtained by other researchers. The rms lift coefficient $t-std(Cl)$ for Wt 0.5 is a bit higher and probably the reason is lower number of time steps during its simulation.

$FilterWeight$	t-avg(Cd)	t-avg(Cl)	t-std(Cl)	St_D
0.5	1.105	-0.043	1.057	0.119
0.1	1.031	-0.010	0.7517	0.120
0.05	0.997	0.037	0.6930	0.108

Table 2: Bulk parameters: numerical results

Source	Re_D	*Appraoch*	$t-avg$(Cd)	$t-avg$(Cl)	$t-std$(Cl)	St_D
Arslan et al.(2011)	$2.64*10^4$	LES	$0.984 \rightarrow 1.39$	-	0.59-0.84	0.107-0.16
Bruno et al.(2011)	$4*10^4$	LES	$0.96 \rightarrow 1.03$	$-0.315to-0.0024$	$0.2 \rightarrow 0.73$	$0.112 \rightarrow 0.122$
Grozescu et al.(2011)	$2*10^4$	VMS-LES	0.96	0.0022	0.35	0.122
Grozescu et al.(2011)	$4*10^4$	VMS-LES	$0.97 \rightarrow 0.98$	$-0.097to0.0043$	$0.52 \rightarrow 0.65$	$0.107 \rightarrow 0.11$
Mannini and Schewe (2011)	$2.64*10^4$	DES	$0.965 \rightarrow 1.016$	$-0.087to0.085$	$0.173 \rightarrow 0.553$	$0.087 \rightarrow 0.119$
Wei and Kareem (2011)	10^5	LES	$1.165 \rightarrow 1.305$	$-0.33to0.42$	$0.495 \rightarrow 1.465$	-

Table 3: Bulk parameters: numerical results

4.2.2 Main flow features

The spanwise-averaged Cp distributions for these three weights are presented in Figures 20, 21,22,23,24 and 25 where Figures 20, 22 and 24 are the Cp of the upper surface of the cylinder and the remaining for the lower surface. The abscissa s denotes the distance from the cylinder leading age measured along the cylinder side and the value of D is 1. Figure 19 consists of the Cp distributions obtained by other investigators. [1]

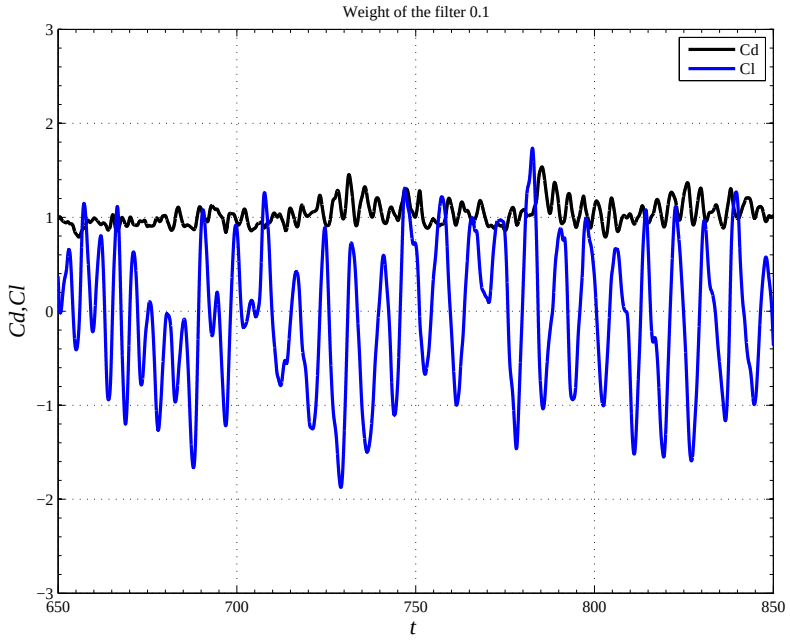


Figure 15: Lift and Drag Coefficients

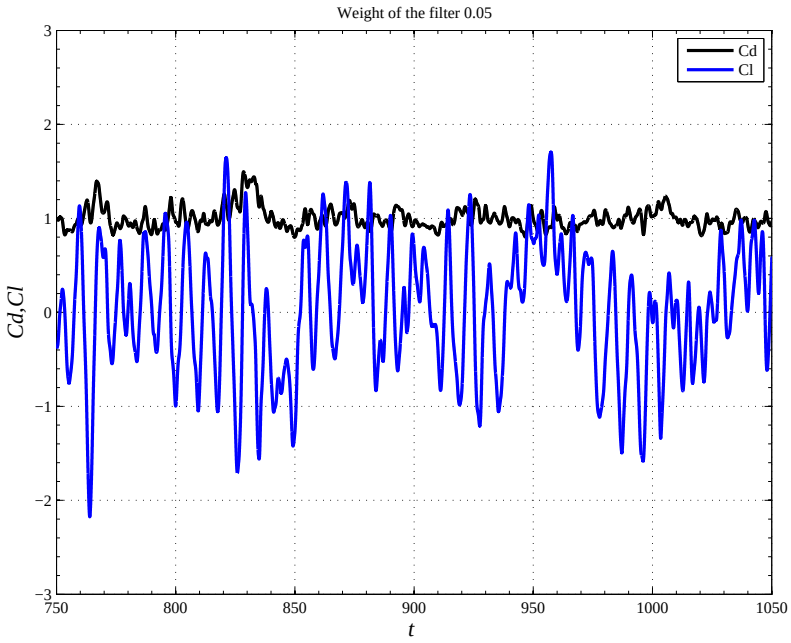


Figure 16: Lift and Drag coefficients

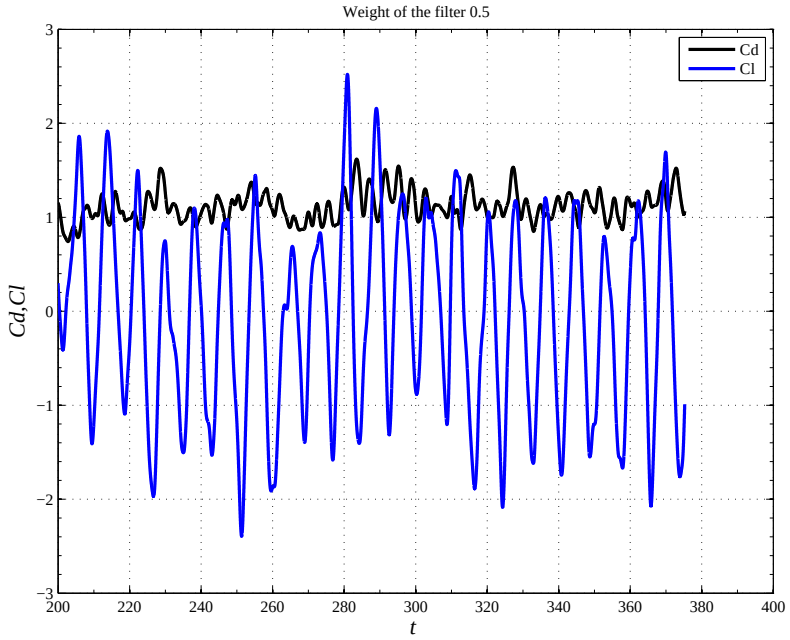


Figure 17: Lift and Drag coefficients

The mean flow is supposed to have a symmetrical behavior in the upper and lower part of the cylinder, so that, the Cp drawn for the upper and lower surface is the same. When the weight of the filter is 0.05, the behavior is practically symmetric but for other two cases it's not exactly. This behavior depends on the time samples used to compare statistics, which is larger for the case with weight equal to 0.05.

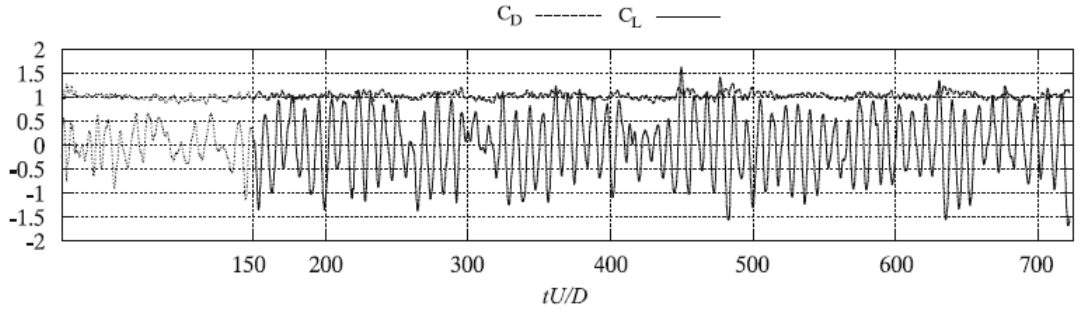


Figure 18: Lift and Drag Coefficients by Bruno(2010)

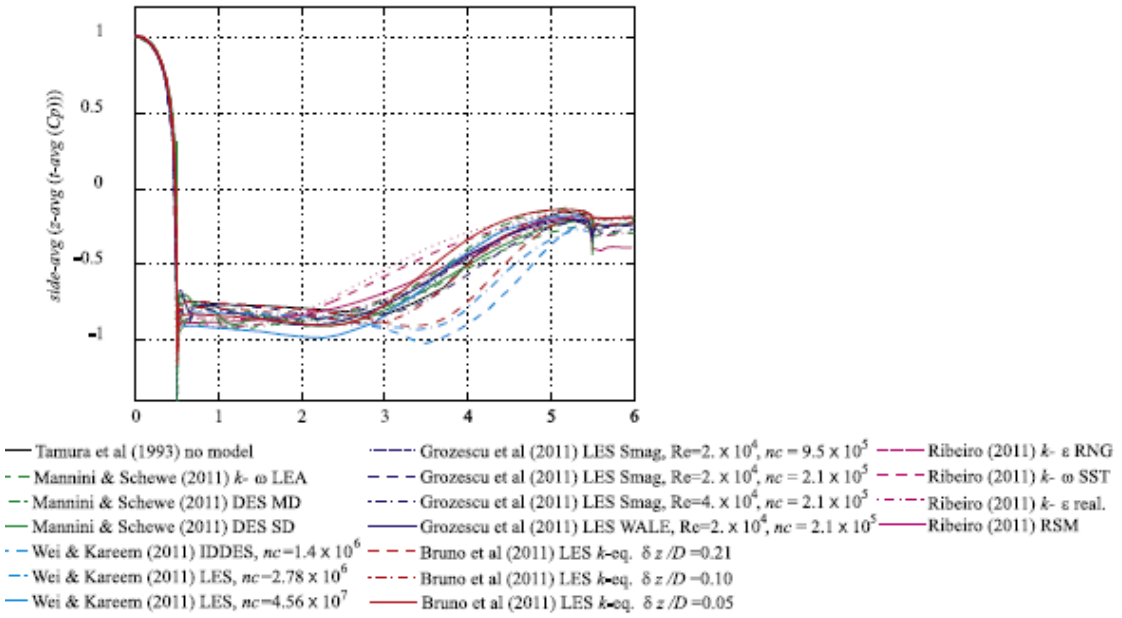


Figure 19: Spanwise averaged Cp distribution: Reference results

Firstly, a first zone of constant low pressure is observed (0-0.5). More downstream, in all cases the pressure increases again because of the change in the curvature of the mean streamlines as the mean flow tends to reattach. It could be observed that the variation between these three cases in this region is obvious (0.5-6) and finally on the base the pressure maintains constant throughout the surface. Practically, the result obtained for these three cases and the results obtained by various investigators using various methods for simulation are similar.

Now, Figures 26,27 and 28 present the spanwise-averaged streamlines for these three cases and Figure 29 contains the upper spanwise-avergaed streamlines ob-

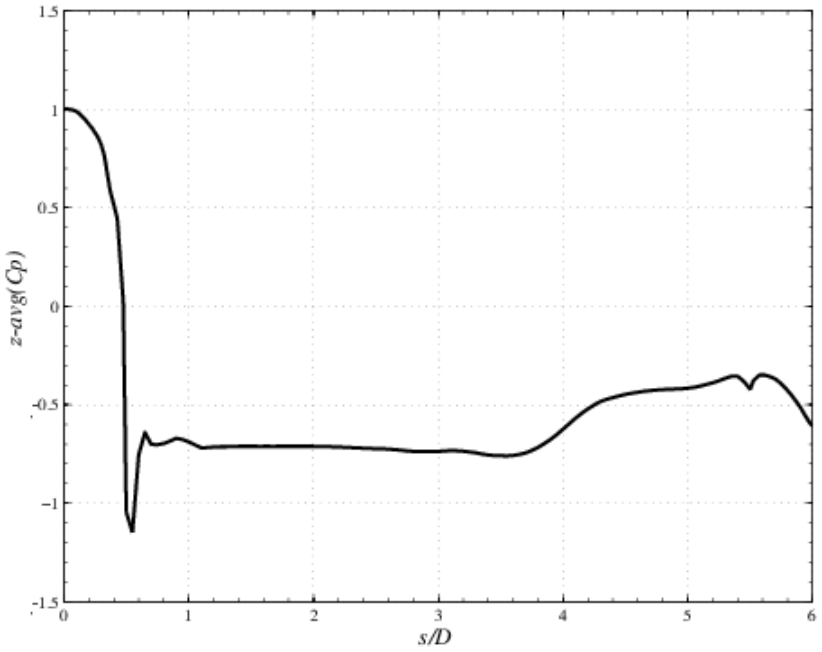


Figure 20: Spanwise averaged Cp distribution for upper surface(Wt-0.5)

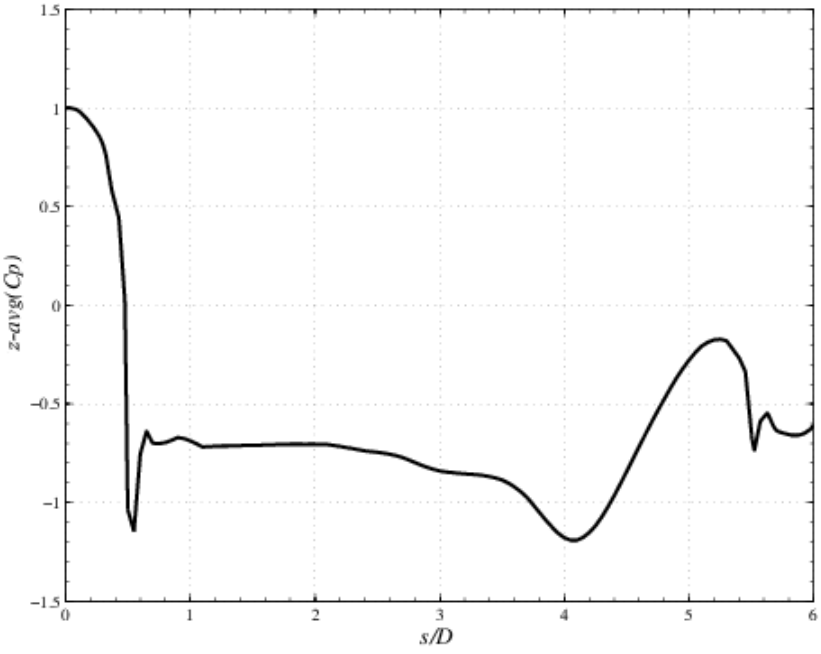


Figure 21: Cp distribution for lower surface(Wt-0.5)

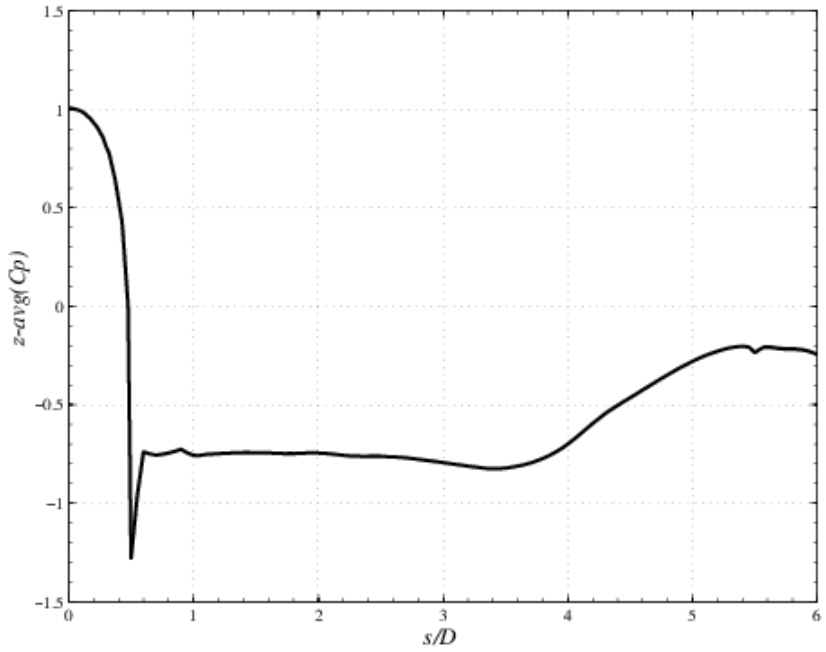


Figure 22: Cp distribution for upper surface(Wt-0.1)

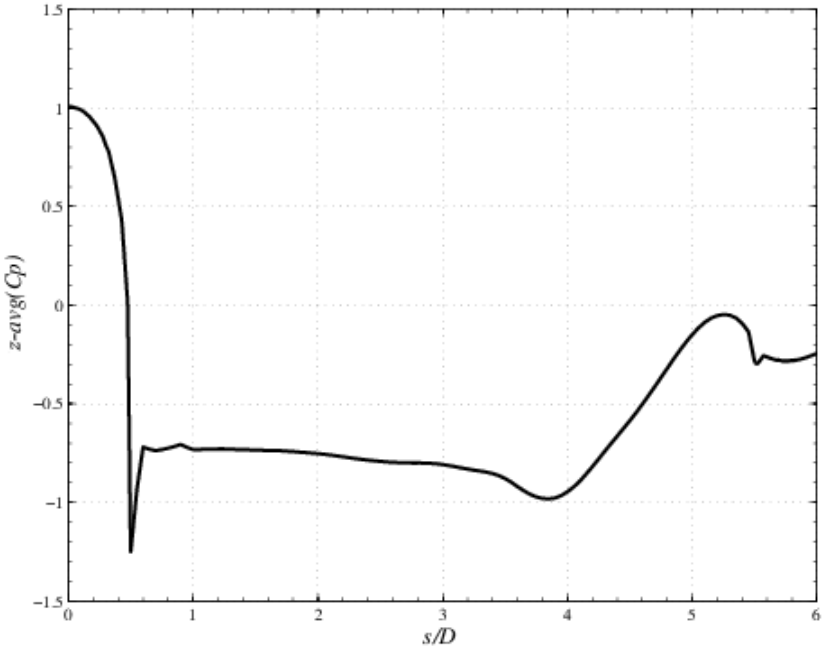


Figure 23: Cp distribution for lower surface(Wt-0.1)

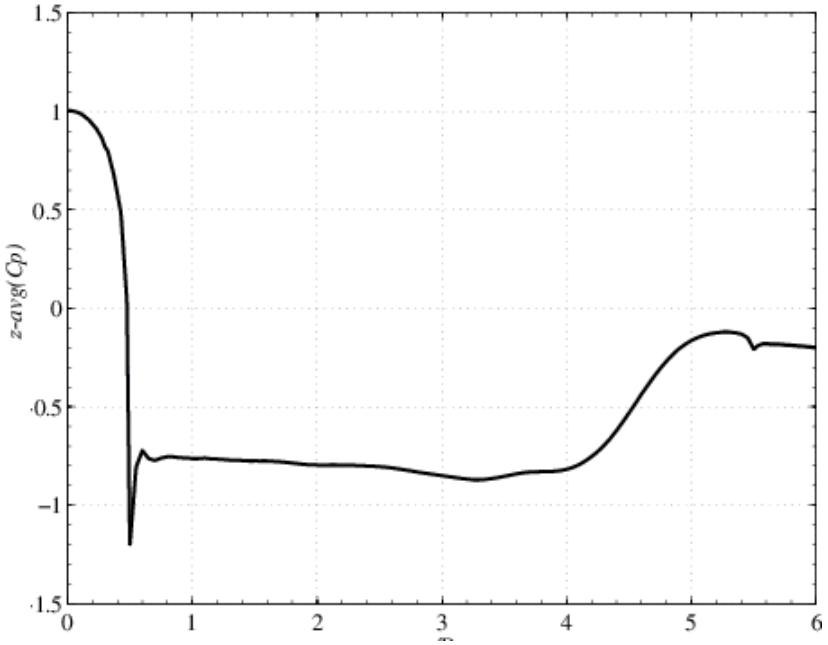


Figure 24: Cp distribution for upper surface(Wt-0.05)

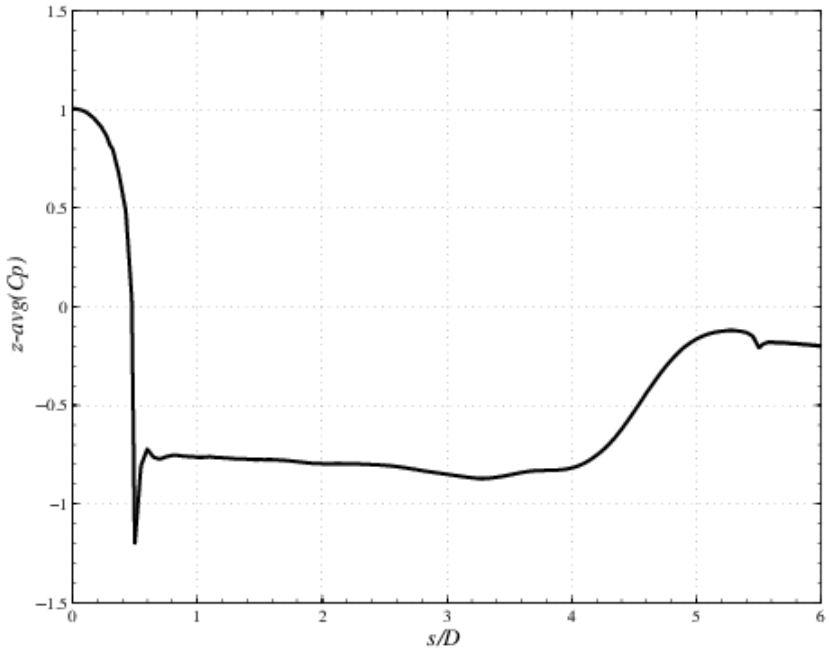


Figure 25: Cp distribution for lower surface(Wt-0.05)

tained by other researchers. The result obtained for lower weight coincides more with other researchers (specially in the last part) so it could be concluded that the result is better for lighter filter than for the heavier ones.
As observed, the mean flow on the cylinder lateral side is characterized by a main recirculation zone for almost all of the cases whose size and shape vary on the variation of the weights of the filter. Moreover it is observed that normally, LES simulations contain more than one recirculation zone in the upper side.

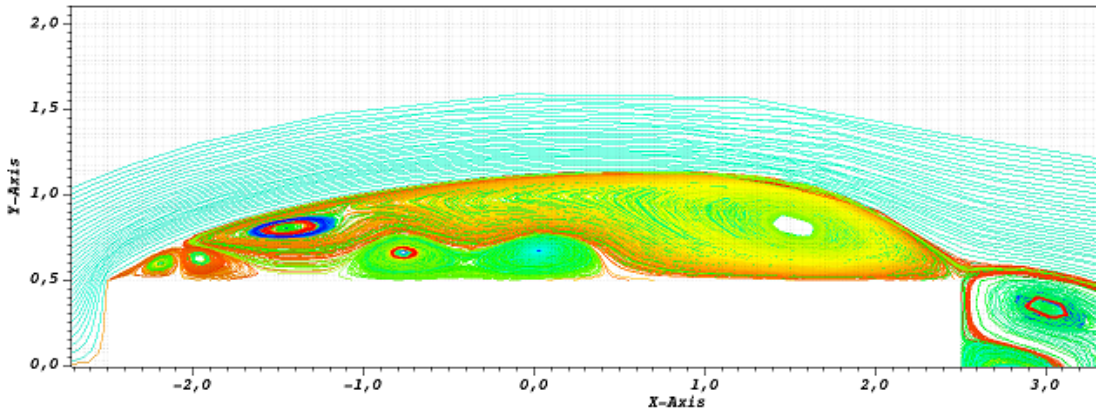


Figure 26: Spanwise-averaged streamlines(Wt:0.5)

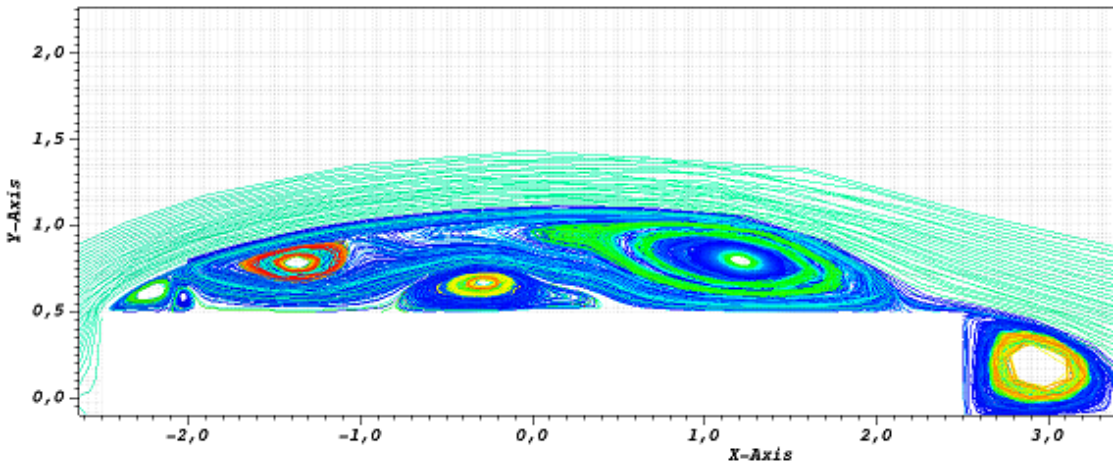


Figure 27: Spanwise-averaged streamlines(Wt:0.1)

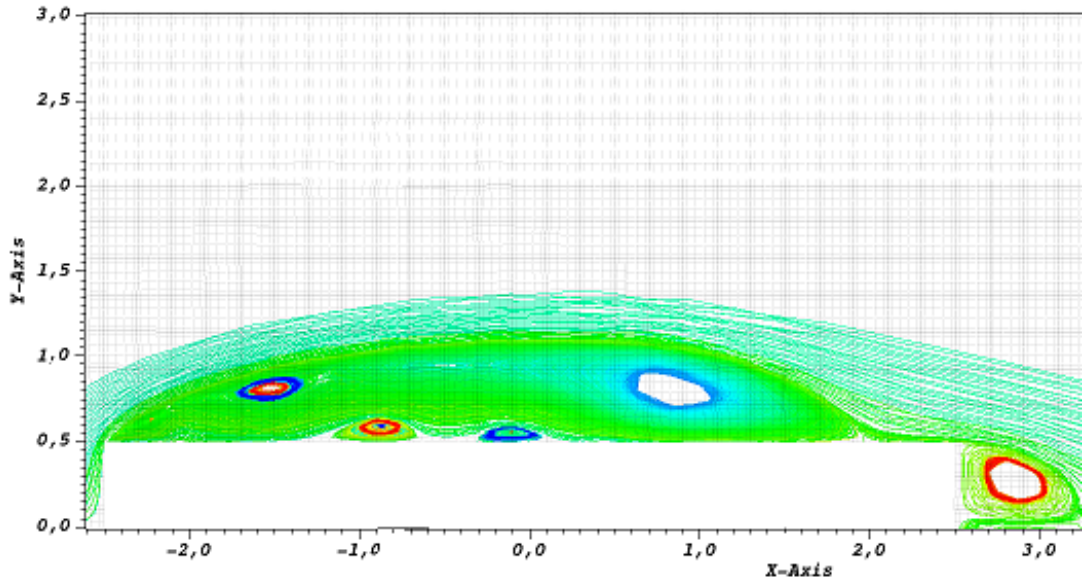


Figure 28: Spanwise-averaged streamlines(Wt:0.05)

$FilterWeight$	x_r	x_c	y_c
0.5	0.8,2.5	-1.96,-1.4,1.55	0.625,0.772,0.78
0.1	-1.8,1.15,2.4	-2.2,-1.35,-0.25,1.25	0.6,0.8,0.65,0.83
0.05	.31,2.15	-1.5,0.8-0.9	0.825,0.778

Table 4: Mean reattachment location and centre of the mean reciruculation zoner for Spanwise-averaged Streamlines

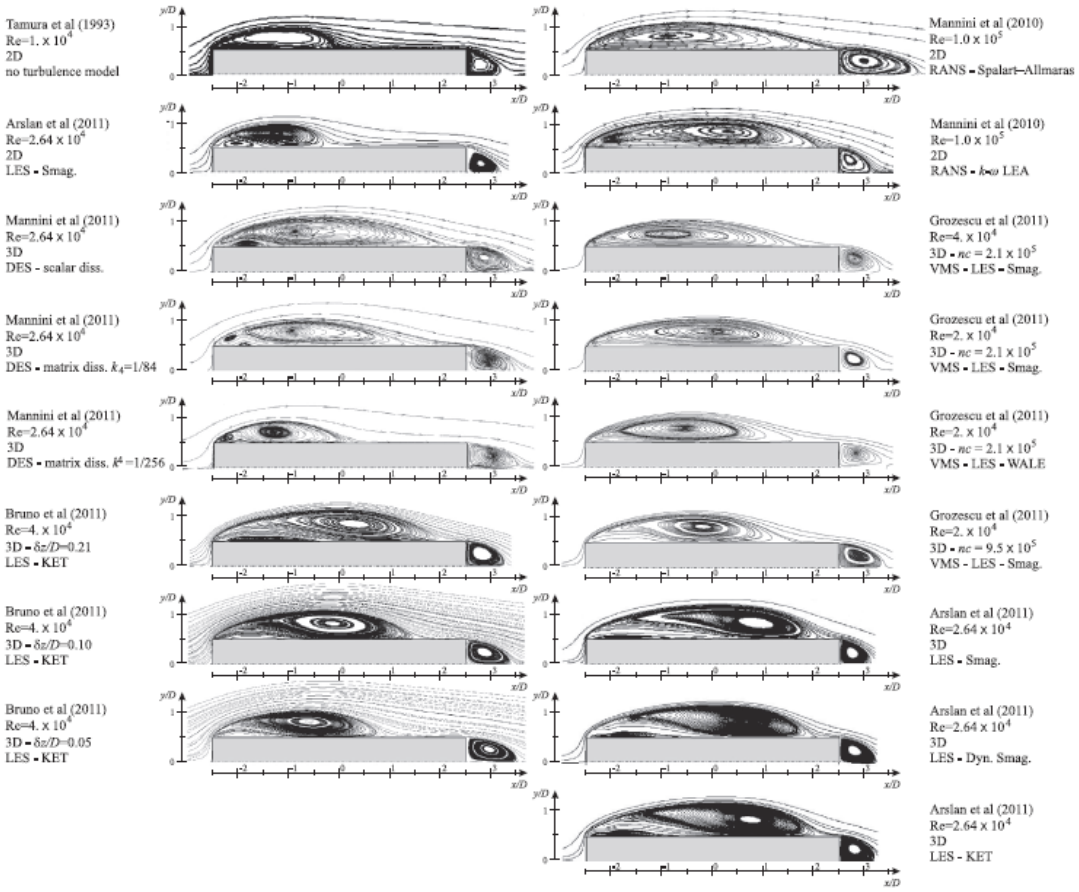


Figure 29: Spanwise-averaged streamlines done by various Investigators

Source	x_r	x_c	y_c
Arslan et al.(2011)	0.66-2.29	-1.24 to 0.75	0.78-0.82
Bruno et al.(2010)	2.18	0.04	0.8
Grozescu et al.(2011 a)	1.65-2.1	-0.97 to 0.09	0.76 to 0.805
Grozescu et al.(2011 b)	1.64	-0.17	0.35-0.82
Mannini et al.(2011)	2.25	-	

Table 5: Mean reattachment location (x_r) and coordinates of the centre of the mean recirculation zone (x_c, y_c)

The main features of the mean flow topology from the Figures 26,27 and 28 are illustrated in Table 4and the result obtained by other researcher are in Table 5. Tables 4 and 5 show tha x coordinate at which the mean recirculation zone ends(mean reattachment location x_r) and the coordinates of the centre of the main recirculation zone (x_c,y_c). [1]
Tables 4 and 5 confirm a significant variability of the length and of the x position of the centre of the mean vortex, while the normal distance from the cylinder to its centre remains almost constant. This leads to different shapes and curvature of the mean streamlines at the edge of recirculation zone (Figure 26).

5 Conclusion

In the present thesis, a preliminary work has been carried out aimed at adapting and validating the open-source spectral-element code NEK500.

First, the capabilities of a simple LES approach, based on explicit filtering of the highest resolved modes without any explicit SGS model, have been appraised for turbulent channel flow. An ad-hoc module was available in NEK5000 ; hence, no code modification was required. Simulations were carried out by varying the number of filtered modes and the filter weight. The obtained results have been compared with those of LES simulations in the literature. It was observed that when filtering 3 modes, for all the considered weights, while the results obtained by filtering 2 or 3 modes were in much better agreement with reference data.

Then, the same LES approach has been applied to the simulation of the BARC test case. A modification of the module used for channel flow was needed because the BARC test case has only a homogeneous direction. A grid was also generated. Simulations were carried out by filtering two modes and by considering different filter weights. Bulk parameters and flow statistics were computed and compared with those obtained in the different contributions to BARC. The general agreement is good and some general remarks can be made. The drag coefficient is rather insensitive to the filter weight, while, as also observed in the BARC benchmark, the lateral loads and the mean flow on the cylinder lateral surfaces significantly vary among the different simulations.

The present work is certainly a good basis for future assessment of the reliability of LES based on the NEK5000 code.

References

[1] Bruno,L., Salvetti, M.V., Ricciardelli, F. *Benchmark on the Aerodynamics of a Rectangular 5:1 Cylinder: an overview after the first four years of activity*

[2] http://www.aniv-iawe.org/barc/docs/BARC_poster.pdf

[3] Grozescu, A.N. *Numerical simulation and control of separated flows*, May, 2012

[4] Rektorys, K., *Variational methods in Mathematics, Science and Engineering*, D. Reidel Publishing Company, Dordrecht-Hollanf/Boston-U.S.A., 2th edition, 1975

[5] Bertóti, E.: *On mixed variational formulation of linear elasticity using non-symmetric stresses and displacements*, International Journal for Numerical Methods in Engineering., **42**, (1997), 561-578.

[6] https://nek5000.mcs.anl.gov/index.php/Main_Page

[7] http://www.scholarpedia.org/article/Turbulence:Subgrid_Scale_Modeling

[8] http://www.arc.vt.edu/ansys_help/flu_th/flu_th_sec_les_sgs_models.html#flu_th_eq_turb_les_sgs_model

[9] Salvetti, M. V. *Modellistica della turbolenza* ,University of Pisa.,(2005-2006)

[10] http://www.mcs.anl.gov/~fischer/Nek5000/primer.pdf

[11] Patera, A.T.: *A spectral element method for fluid dynamics - Laminar flow in a channel expansion.*, Journal of Computational Physics,54:468-488,1984.

[12] https://nek5000.mcs.anl.gov/index.php/UG

[13] https://nek5000.mcs.anl.gov/index.php/Genbox

[14] http://www.mcs.anl.gov/research/projects/nek5/docs/html/navier5_8f.html

[15] Bruno,L., Frabsis,D., Coste, N., Bosco,A. *3D flow around a rectangular cylinder: A computational study*, Politecnico di Torino., (2010)

A Appendix: .box file

It is important to understand that the .box files are used to generate the .rea file. The mesh obtained from the box file presented below are presented in Figure 7. One of the important things to understand about creating a mesh with a lots of boxes in NEK5000 is that the number of elements in each axis must coincide. In this mesh, boxes 1-6 are located in same y-axis and for that reason the number of elements in y-direction (nely) is same for all the 6 boxes (also for boxes 7-12, 13-18, 19-24, 25-30). Moreover, Boxes 1,7,13,19 and 25 are situated in the same x-axis (also boxes 2-8-....) and so that all these boxes have same number of elements in x-direction (nelx). This is the main reason for being unable to design a mesh like Figure 9.
Probably it could be noticed that there is no Box 13 as it is the place where the 5:1 cylinder is placed. The boundary condition that limits the cylinder is kept as wall condition so that the hole acts as a cylinder.
Remember that after modifying the .box file, the command genbox should be used to generate the .rea file and after that the box.rea file generated must be renamed. For example, in this case there is a directory called turbBarc that contains all the files necessary to be run in NEK5000 (turbBarc.rea, turbBarc.usr, turbBarc.box, SIZE file etc.). Once the .box file is modified, doing genbox a new rea file called box.rea is created. This file must be renamed to turbBarc.rea file and once finished genmap should be used to create a new .map file.In order to understand the box file very detailed, it is highly recommended to visit the official box file page (look [13]).

```
turbChannel.rea
-2                      spatial dimension  ( < 0 --> generate .rea/.re2 pair)
1                      number of fields
#=======================================================================
#
#    Example of .box file for Taylor-Green
#
#    If nelx (y or z) < 0, then genbox automatically generates the
#                          grid spacing in the x (y or z) direction
#                          with a geometric ratio given by "ratio".
#                          ( ratio=1 implies uniform spacing )
#
#    Note that the character bcs _must_ have 3 spaces.
#
#=======================================================================
#####
Box 1
-15  -15                                          nelx,nely,nelz for Box
-75 -4.5 0.73                                     x0,x1,gain  (rescaled in usrdat)
-75.5 -2.5 0.7407                                 y0,y1,gain  (rescaled in usrdat)
#0 5 1.                                           z0,z1,gain  (rescaled in usrdat)
v  ,   ,O  ,                                      bc's  (3 chars each!)
#####
Box 2
-15  -15                                          nelx,nely,nelz for Box
-4.5 -2.5 .926                                    x0,x1,gain  (rescaled in usrdat)
-75.5 -2.5 0.7407                                 y0,y1,gain  (rescaled in usrdat)
#0 5 1.                                           z0,z1,gain  (rescaled in usrdat)
   ,   ,O  ,                                      bc's  (3 chars each!)
#####
Box 3
-70  -15                                          nelx,nely,nelz for Box
-2.5 2.5 1                                        x0,x1,gain  (rescaled in usrdat)
-75.5 -2.5 0.7407                                 y0,y1,gain  (rescaled in usrdat)
#0 5 1.                                           z0,z1,gain  (rescaled in usrdat)
   ,   ,O  ,                                      bc's  (3 chars each!)
#####
Box 4
-15  -15                                          nelx,nely,nelz for Box
2.5 4.5 1.079                                     x0,x1,gain  (rescaled in usrdat)
-75.5 -2.5 0.7407                                 y0,y1,gain  (rescaled in usrdat)
#0 5 1.                                           z0,z1,gain  (rescaled in usrdat)
   ,   ,O  ,                                      bc's  (3 chars each!)
#####
Box 5
-20  -15                                          nelx,nely,nelz for Box
4.5 10.5 1.03                                     x0,x1,gain  (rescaled in usrdat)
-75.5 -2.5 0.7407                                 y0,y1,gain  (rescaled in usrdat)
#0 5 1.                                           z0,z1,gain  (rescaled in usrdat)
   ,   ,O  ,                                      bc's  (3 chars each!)
#####
Box 6
-25  -15                                          nelx,nely,nelz for Box
10.5 125 1.2                                      x0,x1,gain  (rescaled in usrdat)
-75.5 -2.5 0.7407                                 y0,y1,gain  (rescaled in usrdat)
#0 5 1.                                           z0,z1,gain  (rescaled in usrdat)
   ,O  ,O  ,                                      bc's  (3 chars each!)
#####
Box 7
-15  -15                                          nelx,nely,nelz for Box
-75 -4.5 0.73                                     x0,x1,gain  (rescaled in usrdat)
-2.5 -.5 .926                                     y0,y1,gain  (rescaled in usrdat)
#0 5 1.                                           z0,z1,gain  (rescaled in usrdat)
```

```
v  ,   ,   ,                          bc's  (3 chars each!)
#####
Box 8
-15  -15                              nelx,nely,nelz for Box
-4.5 -2.5 .926                       x0,x1,gain  (rescaled in usrdat)
-2.5 -.5 .926                        y0,y1,gain  (rescaled in usrdat)
#0 5 1.                              z0,z1,gain  (rescaled in usrdat)
   ,   ,   ,                         bc's  (3 chars each!)
#####
Box 9
-70  -15                             nelx,nely,nelz for Box
-2.5 2.5 1                           x0,x1,gain  (rescaled in usrdat)
-2.5 -.5 .926                        y0,y1,gain  (rescaled in usrdat)
#0 5 1.                              z0,z1,gain  (rescaled in usrdat)
   ,   ,   ,W                        bc's  (3 chars each!)
#####
Box 10
-15  -15                             nelx,nely,nelz for Box
2.5 4.5 1.079                        x0,x1,gain  (rescaled in usrdat)
-2.5 -.5 .926                        y0,y1,gain  (rescaled in usrdat)
#0 5 1.                              z0,z1,gain  (rescaled in usrdat)
   ,   ,   ,                         bc's  (3 chars each!)
#####
Box 11
-20  -15                             nelx,nely,nelz for Box
4.5 10.5 1.03                        x0,x1,gain  (rescaled in usrdat)
-2.5 -.5 .926                        y0,y1,gain  (rescaled in usrdat)
#0 5 1.                              z0,z1,gain  (rescaled in usrdat)
   ,   ,   ,                         bc's  (3 chars each!)
#####
Box 12
-25  -15                             nelx,nely,nelz for Box
10.5 125 1.2                         x0,x1,gain  (rescaled in usrdat)
-2.5 -.5 .926                        y0,y1,gain  (rescaled in usrdat)
#0 5 1.                              z0,z1,gain  (rescaled in usrdat)
   ,O  ,   ,                         bc's  (3 chars each!)
#####
Box 13
-15  -14                             nelx,nely,nelz for Box
-75 -4.5 0.73                        x0,x1,gain  (rescaled in usrdat)
-.5 .5 1                             y0,y1,gain  (rescaled in usrdat)
#0 5 1.                              z0,z1,gain  (rescaled in usrdat)
v  ,   ,   ,                         bc's  (3 chars each!)
#####
Box 14
-15  -14                             nelx,nely,nelz for Box
-4.5 -2.5 .926                       x0,x1,gain  (rescaled in usrdat)
-.5 .5 1                             y0,y1,gain  (rescaled in usrdat)
#0 5 1.                              z0,z1,gain  (rescaled in usrdat)
   ,W  ,   ,                         bc's  (3 chars each!)
#####
Box 16
-15  -14                             nelx,nely,nelz for Box
2.5 4.5 1.079                        x0,x1,gain  (rescaled in usrdat)
-.5 .5 1                             y0,y1,gain  (rescaled in usrdat)
#0 5 1.                              z0,z1,gain  (rescaled in usrdat)
W  ,   ,   ,                         bc's  (3 chars each!)
#####
Box 17
-20  -14                             nelx,nely,nelz for Box
4.5 10.5 1.03                        x0,x1,gain  (rescaled in usrdat)
-.5 .5 1                             y0,y1,gain  (rescaled in usrdat)
#0 5 1.                              z0,z1,gain  (rescaled in usrdat)
```

```
   ,   ,   ,                                bc's  (3 chars each!)
#####
Box 18
-25  -14                                    nelx,nely,nelz for Box
10.5 125 1.2                               x0,x1,gain  (rescaled in usrdat)
-.5 .5 1                                   y0,y1,gain  (rescaled in usrdat)
#0 5 1.                                    z0,z1,gain  (rescaled in usrdat)
   ,O  ,   ,
#####
Box 19
-15  -15                                    nelx,nely,nelz for Box
-75 -4.5 0.73                              x0,x1,gain  (rescaled in usrdat)
.5 2.5 1.079                               y0,y1,gain  (rescaled in usrdat)
#0 5 1.                                    z0,z1,gain  (rescaled in usrdat)
v  ,   ,   ,                               bc's  (3 chars each!)
#####
Box 20
-15  -15                                    nelx,nely,nelz for Box
-4.5 -2.5 .926                             x0,x1,gain  (rescaled in usrdat)
.5 2.5 1.079                               y0,y1,gain  (rescaled in usrdat)
#0 5 1.                                    z0,z1,gain  (rescaled in usrdat)
   ,   ,   ,                               bc's  (3 chars each!)
#####
Box 21
-70  -15                                    nelx,nely,nelz for Box
-2.5 2.5 1                                 x0,x1,gain  (rescaled in usrdat)
0.5 2.5 1.079                              y0,y1,gain  (rescaled in usrdat)
#0 5 1.                                    z0,z1,gain  (rescaled in usrdat)
   ,   ,W  ,                               bc's  (3 chars each!)
#####
Box 22
-15  -15                                    nelx,nely,nelz for Box
2.5 4.5 1.079                              x0,x1,gain  (rescaled in usrdat)
.5 2.5 1.079                               y0,y1,gain  (rescaled in usrdat)
#0 5 1.                                    z0,z1,gain  (rescaled in usrdat)
   ,   ,   ,                               bc's  (3 chars each!)
#####
Box 23
-20  -15                                    nelx,nely,nelz for Box
4.5 10.5 1.03                              x0,x1,gain  (rescaled in usrdat)
.5 2.5 1.079                               y0,y1,gain  (rescaled in usrdat)
#0 5 1.                                    z0,z1,gain  (rescaled in usrdat)
   ,   ,   ,                               bc's  (3 chars each!)
#####
Box 24
-25  -15                                    nelx,nely,nelz for Box
10.5 125 1.2                               x0,x1,gain  (rescaled in usrdat)
0.5 2.5 1.079                              y0,y1,gain  (rescaled in usrdat)
#0 5 1.                                    z0,z1,gain  (rescaled in usrdat)
   ,O  ,   ,                               bc's  (3 chars each!)
#####
Box 25
-15  -15                                    nelx,nely,nelz for Box
-75 -4.5 0.73                              x0,x1,gain  (rescaled in usrdat)
2.5 75.5 1.35                              y0,y1,gain  (rescaled in usrdat)
#0 5 1.                                    z0,z1,gain  (rescaled in usrdat)
v  ,   ,   ,O                              bc's  (3 chars each!)
#####
Box 26
-15  -15                                    nelx,nely,nelz for Box
-4.5 -2.5 .926                             x0,x1,gain  (rescaled in usrdat)
2.5 75.5 1.35                              y0,y1,gain  (rescaled in usrdat)
#0 5 1.                                    z0,z1,gain  (rescaled in usrdat)
```

```
   ,   ,   ,O                          bc's  (3 chars each!)
#####
Box 27
-70  -15                               nelx,nely,nelz for Box
-2.5 2.5 1                            x0,x1,gain  (rescaled in usrdat)
2.5 75.5 1.35                         y0,y1,gain  (rescaled in usrdat)
#0 5 1.                               z0,z1,gain  (rescaled in usrdat)
   ,   ,   ,O                         bc's  (3 chars each!)
#####
Box 28
-15  -15                              nelx,nely,nelz for Box
2.5 4.5 1.079                         x0,x1,gain  (rescaled in usrdat)
2.5 75.5 1.35                         y0,y1,gain  (rescaled in usrdat)
#0 5 1.                               z0,z1,gain  (rescaled in usrdat)
   ,   ,   ,O                         bc's  (3 chars each!)
##########
Box 29
-20  -15                              nelx,nely,nelz for Box
4.5 10.5 1.03                         x0,x1,gain  (rescaled in usrdat)
2.5 75.5 1.35                         y0,y1,gain  (rescaled in usrdat)
#0 5 1.                               z0,z1,gain  (rescaled in usrdat)
   ,   ,   ,O                         bc's  (3 chars each!)
#####
Box 30
-25  -15                              nelx,nely,nelz for Box
10.5 125 1.2                          x0,x1,gain  (rescaled in usrdat)
2.5 75.5 1.35                         y0,y1,gain  (rescaled in usrdat)
#0 5 1.                               z0,z1,gain  (rescaled in usrdat)
   ,O  ,   ,O                         bc's  (3 chars each!)

(END)
```

The box file for the preliminary verification is posted below. Only 8 boxes are used, in order to have less number of elements possible and to run the simulation as fast as possible.

```
turbChannel.rea
2                     spatial dimension  ( < 0 --> generate .rea/.re2 pair)
1                     number of fields
#=======================================================================
#
#    Example of .box file for Taylor-Green
#
#    If nelx (y or z) < 0, then genbox automatically generates the
#                          grid spacing in the x (y or z) direction
#                          with a geometric ratio given by "ratio".
#                          ( ratio=1 implies uniform spacing )
#
#    Note that the character bcs _must_ have 3 spaces.
#
#=======================================================================
#####
Box 1
-10 -15                                          nelx,nely,nelz for Box
-75 0 .85                                           x0,x1,gain  (rescaled in usrdat)
-75.5 -.5 .85                                        y0,y1,gain  (rescaled in usrdat)
v  ,   ,O  ,                                   bc's  (3 chars each!)
#####
Box 2
-5  -15                                             nelx,nely,nelz for Box
0 5 1.                                                x0,x1,gain  (rescaled in usrdat)
-75.5 -.5 .85                                        y0,y1,gain  (rescaled in usrdat)
   ,   ,O  ,W                                  bc's  (3 chars each!)
#####
Box 3
-20  -15                                          nelx,nely,nelz for Box
5 125 1.15                                            x0,x1,gain  (rescaled in usrdat)
-75.5 -.5 .85                                         y0,y1,gain  (rescaled in usrdat)
   ,O  ,O  ,                                   bc's  (3 chars each!)
#####
Box 4
-10  -1                                  nelx,nely,nelz for Box
-75 0 .85                                             x0,x1,gain  (rescaled in usrdat)
-.5 .5 1.                                               y0,y1,gain  (rescaled in usrdat)
v  ,W  ,   ,                                   bc's  (3 chars each!)
#####
Box 6
-20  -1                                     nelx,nely,nelz for Box
5 125 1.15                                            x0,x1,gain  (rescaled in usrdat)
-.5 .5 1.                                               y0,y1,gain  (rescaled in usrdat)
W  ,O  ,   ,                                 bc's  (3 chars each!)
#####
Box 7
-10  -15                               nelx,nely,nelz for Box
-75 0 .85                                             x0,x1,gain  (rescaled in usrdat)
.5 75.5 1.15                                             y0,y1,gain  (rescaled in usrdat)
v  ,   ,   ,O                                  bc's  (3 chars each!)
#####
Box 8
-5  -15                                           nelx,nely,nelz for Box
0 5 1.                                                x0,x1,gain  (rescaled in usrdat)
.5 75.5 1.15                                             y0,y1,gain  (rescaled in usrdat)
   ,   ,W  ,O                                  bc's  (3 chars each!)
#####
Box 9
-20  -15                                 nelx,nely,nelz for Box
5 125 1.15                                            x0,x1,gain  (rescaled in usrdat)
.5 75.5 1.15                                             y0,y1,gain  (rescaled in usrdat)
   ,O  ,   ,O                                bc's  (3 chars each!)
(END)
```

B Appendix: .rea file

The .rea file is the most preliminary part for the NEK5000 coding. The Reynolds number(p 02), number of iterations (p 11), number of the modes of the filter (p 101), weight of the filter(p 103) are assigned here. During the validation of the turbChannel, it is found that the number of modes assigned to a filter, in reality, is the number of modes assigned in the .rea file plus 1. So if the p101 is assigned 1 then in reality 2 number of modes is used. Moreover if it is wished to restart the simulation from a certain file then in the PRESOLVE/RESTART OPTIONS it could be placed 1 and just below this line the name of the restart file could be placed (in this case restart.f00001).

```
****** PARAMETERS *****
   2.610000     NEKTON VERSION
         2 DIMENSIONAL RUN
        103 PARAMETERS FOLLOW
      1.000000         p01 DENSITY
     -40000.00         p02 VISCOS
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00     p07 RHOCP
     0.0000000E+00     p08 CONDUCT
     0.0000000E+00
     0.0000000E+00     p10 FINTIME
     280000.00         p11 NSTEPS
    -.10000000E-02     p12 DT
     0.0000000E+00     p13 IOCOMM
     0.0000000E+00     p14 IOTIME
     500.000           p15 IOSTEP
     0.0000000E+00     p16 PSSOLVER
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00
     1.0000000E-05     p21 DIVERGENCE
     1.0000000E-08     p22 HELMHOLTZ
 0              p23 NPSCAL
     1.0000000E-02     p24 TOLREL
     1.0000000E-02     p25 TOLABS
      2.000000         p26 COURANT/NTAU
      3.000000         p27 TORDER
     0.0000000E+00     p28 TORDER: mesh velocity (0: p28=p27)
     0.0000000E+00     p29 magnetic visc if > 0, = -1/Rm if < 0
     0.0000000E+00     p30 > 0 ==> properties set in uservp()
     0.0000000E+00     p31 NPERT: #perturbation modes
     0.0000000E+00     p32 #BCs in re2 file, if > 0
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00     p42 0=gmres/1=pcg
     0.0000000E+00     p43 0=semg/1=schwarz
     0.0000000E+00     p44 0=E-based/1=A-based prec.
     0.0000000E+00     p45 Relaxation factor for DTFS
     0.0000000E+00      p46 if >0, dont call SETICS
     0.0000000E+00     p47 vnu: mesh matieral prop
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00     p52 IOHIS
     0.0000000E+00
     0.0000000E+00     p54 1,2,3-->fixed flow rate dir=x,y,z
     0.0000000E+00     p55 vol.flow rate(p54>0) or Ubar(p54<0)
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00
     0.0000000E+00     p59 !=0 --> use std axhelm for all elem
     0.0000000E+00     p60 !=0--> init velocity to small nonzero
     0.0000000E+00
```

```
      0.0000000E+00     p62 >0 --> force byte_swap for output
      0.0000000E+00     p63 =8 --> force 8-byte output
      0.0000000E+00     p64 =1 --> perturbation restart
       1.000000         p65 #iofile(eg,0 or 64); <0 --> sep. dirs
       6.000000         p66 write fmt:ONLY postx uses rea value
       6.000000         p67 read fmt: same modes as p66
       200000.000            p68 iastep: freq for avg_all
      0.0000000E+00
      0.0000000E+00
      0.0000000E+00
      0.0000000E+00
      0.0000000E+00
      0.0000000E+00     p74 verbose Helmholtz
      0.0000000E+00
      0.0000000E+00
      0.0000000E+00
      0.0000000E+00
      0.0000000E+00
      0.0000000E+00
      0.0000000E+00
      0.0000000E+00
      0.0000000E+00
      0.0000000E+00     p84 !=0 -->sets initial timestep if p12>0
      0.0000000E+00     p85 dt ratio if p84 !=0, for timesteps>0
      0.0000000E+00     p86 reserved
      0.0000000E+00
      0.0000000E+00
      0.0000000E+00     p89 reserved
      0.0000000E+00
      0.0000000E+00
      0.0000000E+00
       20.00000         p93 Numbr of prev pressure solns saved
       5.000000         p94 start projecting vel. after p94 step
       5.000000         p95 start projecting pr after p95 step
      0.0000000E+00
      0.0000000E+00
      0.0000000E+00
       4.000000         p99    dealiasing:if <0 disable
      0.0000000E+00     p100 reserved
       2.000000         p101   No. of additional filter modes
      0.0000000E+00
      5.0000001E-01     p103   weight of stabilizing filter (.01)
            0  SCALAR DATA FOLLOW (CONDUCT; RHOCP)
            4  LOGICAL SWITCHES FOLLOW
  T     IFFLOW
  T     IFTRAN
  T     IFADVC (one entry per field)
  T     IFCHAR
   8.00000   8.00000  -0.500000   -4.00000     XFAC,YFAC,XZERO,YZERO
 **MESH DATA** 6 lines are X,Y,Z;X,Y,Z. Columns corners 1-4;5-8
      -10860  2       10860           NEL,NDIM,NELV
            1 PRESOLVE/RESTART OPTIONS  *****
restart.f00001
          7         INITIAL CONDITIONS *****
C Default
C Default
C Default
C Default
C Default
C Default
C Default
  ***** DRIVE FORCE DATA ***** BODY FORCE, FLOW, Q
            4                 Lines of Drive force data follow
C
C
```

```
C
C
  ***** Variable Property Data ***** Overrrides Parameter data.
            1 Lines follow.
            0 PACKETS OF DATA FOLLOW
  ***** HISTORY AND INTEGRAL DATA *****
            0   POINTS.  Hcode, I,J,H,IEL
  ***** OUTPUT FIELD SPECIFICATION *****
            6 SPECIFICATIONS FOLLOW
  F      COORDINATES
  T      VELOCITY
  T      PRESSURE
  F      TEMPERATURE
  F      TEMPERATURE GRADIENT
  0      PASSIVE SCALARS
  ***** OBJECT SPECIFICATION *****
       0 Surface Objects
       0 Volume  Objects
       0 Edge    Objects
       0 Point   Objects
(END)
```

C Appendix: SIZE file

As the name itself explains, the size of the parameters are defined here. One of the most important terms to be understood of the SIZE file is the parameters *lelt, lp and lelg*. The parameter *lelg*is the total number of elements generated during the box creation (it also could be seen in the .rea file), *lp* is the number of processors used to simulate and *lelt* is the number of elements for each processor i.e. $lelt \geq \frac{lelg}{lp}$.

```
C     Dimension file to be included
C
C     HCUBE array dimensions
C
      parameter (ldim=2)
      parameter (lx1=6,ly1=lx1,lz1=1,lelt=1500,lelv=lelt)
      parameter (lxd=9,lyd=lxd,lzd=1)
      parameter (lelx=1,lely=lelx,lelz=lelx)

      parameter (lzl=3 + 2*(ldim-3))

      parameter (lx2=lx1)
      parameter (ly2=lx1)
      parameter (lz2=1)
      parameter (lx3=lx2)
      parameter (ly3=ly2)
      parameter (lz3=lz2)

      parameter (lp = 8)
      parameter (lelg = 12000)
c
c     parameter (lpelv=lelv,lpelt=lelt,lpert=3)  ! perturbation
c     parameter (lpx1=lx1,lpy1=ly1,lpz1=lz1)     ! array sizes
c     parameter (lpx2=lx2,lpy2=ly2,lpz2=lz2)
c
      parameter (lpelv=1,lpelt=1,lpert=1)        ! perturbation
      parameter (lpx1=1,lpy1=1,lpz1=1)           ! array sizes
      parameter (lpx2=1,lpy2=1,lpz2=1)
c
c     parameter (lbelv=lelv,lbelt=lelt)          ! MHD
c     parameter (lbx1=lx1,lby1=ly1,lbz1=lz1)     ! array sizes
c     parameter (lbx2=lx2,lby2=ly2,lbz2=lz2)
c
      parameter (lbelv=1,lbelt=1)                ! MHD
      parameter (lbx1=1,lby1=1,lbz1=1)           ! array sizes
      parameter (lbx2=1,lby2=1,lbz2=1)

C     LX1M=LX1 when there are moving meshes; =1 otherwise
      parameter (lx1m=1,ly1m=1,lz1m=1)
      parameter (ldimt= 3)                       ! passive scalars + T
      parameter (ldimt1=ldimt+1)
      parameter (ldimt3=ldimt+3)
c
c     Note:  In the new code, LELGEC should be about sqrt(LELG)
c
      PARAMETER (LELGEC = 1)
      PARAMETER (LXYZ2  = 1)
      PARAMETER (LXZ21  = 1)

      PARAMETER (LMAXV=LX1*LY1*LZ1*LELV)
      PARAMETER (LMAXT=LX1*LY1*LZ1*LELT)
      PARAMETER (LMAXP=LX2*LY2*LZ2*LELV)
      PARAMETER (LXZ=LX1*LZ1)
      PARAMETER (LORDER=3)
      PARAMETER (MAXOBJ=4,MAXMBR=LELT*6)
      PARAMETER (lhis=100)         ! # of pts a proc reads from hpts.in
                                   ! Note: lhis*np > npoints in hpts.in
C
C     Common Block Dimensions
C
      PARAMETER (LCTMP0 =2*LX1*LY1*LZ1*LELT)
      PARAMETER (LCTMP1 =4*LX1*LY1*LZ1*LELT)
C
C     The parameter LVEC controls whether an additional 42 field arrays
C     are required for Steady State Solutions.  If you are not using
```

```
C     Steady State, it is recommended that LVEC=1.
C
      PARAMETER (LVEC=1)
C
C     Uzawa projection array dimensions
C
      parameter (mxprev = 20)
      parameter (lgmres = 40)
C
C     Split projection array dimensions
C
      parameter(lmvec = 1)
      parameter(lsvec = 1)
      parameter(lstore=lmvec*lsvec)
c
c     NONCONFORMING STUFF
c
      parameter (maxmor = lelt)
C
C     Array dimensions
C
      COMMON/DIMN/NELV,NELT,NX1,NY1,NZ1,NX2,NY2,NZ2
     $,NX3,NY3,NZ3,NDIM,NFIELD,NPERT,NID
     $,NXD,NYD,NZD

c automatically added by makenek
      parameter(lxo   = lx1) ! max output grid size (lxo>=lx1)

c automatically added by makenek
      parameter(lpart = 1E7  ) ! max number of particles

c automatically added by makenek
      integer ax1,ay1,az1,ax2,ay2,az2
      parameter (ax1=lx1,ay1=ly1,az1=lz1,ax2=lx2,ay2=ly2,az2=lz2) ! running averages

c automatically added by makenek
      parameter (lxs=1,lys=lxs,lzs=(lxs-1)*(ldim-2)+1) !New Pressure Preconditioner

c automatically added by makenek
      parameter (lfdm=0)  ! == 1 for fast diagonalization method

c automatically added by makenek
      common/IOFLAG/nio  ! for logfile verbosity control
(END)
```

D Appendix: .usr file

The subroutine my_avg_all contains all the post processing code and the running averages $E(X)$, $E(X^2)$,$E(X * Y)$. First of all, the parameters necessary are defined and after checking that the parameters have the required size, the force averages and the running averages are calculated and the obtained results are saved in the file "Force_aver.dat". Remember that the command *call outpost2* outputs the average fields so that the result could be visualized by visualization tools like *Fluent, Paraview etc.*.
Finally, the the z_average functions is used to calculate the running averages like in 2D flow and the rms velocities are calculated and are out-posted.

```
c-----------------------------------------------------------------------
       subroutine my_avg_all
c
c     This routine computes running averages E(X),E(X^2),E(X*Y)
c     and outputs to avg*.fld*, rms*.fld*, and rm2*.fld* for all
c     fields.
c
c     E denotes the expected value operator and X,Y two
c     real valued random variables.
c
c     variances and covariances can be computed in a post-processing step:
c
c        var(X)   := E(X^X) - E(X)*E(X)
c        cov(X,Y) := E(X*Y) - E(X)*E(Y)
c
c     Note: The E-operator is linear, in the sense that the expected
c           value is given by E(X) = 1/N * sum[ E(X)_i ], where E(X)_i
c           is the expected value of the sub-ensemble i (i=1...N).
c
      include 'SIZE'
      include 'TOTAL'
      include 'AVG'

      integer icalld
      save    icalld
      data    icalld  /0/
      integer i1,i2,i3,i4
      integer i11,i21,i31,i41

      parameter (l2d=lx1*ly1*lelx) ! Number of points in 2D field
      common /mystuff/ u2d(l2d),v2d(l2d)

      common /scravg/ ua(lx1*ly1*lelx*lely)
     $              , va(lx1*ly1*lelx*lely)
     $              , wa(lx1*ly1*lelx*lely)
     $              , ua1(lx1*ly1*lelx*lely)
     $              , va1(lx1*ly1*lelx*lely)
     $              , wa1(lx1*ly1*lelx*lely)
     $              , w1(lx1*ly1*lelx*lely)
     $              , w2(lx1*ly1*lelx*lely)
     $              , pa(lx2*ly2*lelx*lely)
     $              , pa1(lx2*ly2*lelx*lely)
     $              , w1p(lx2*ly2*lelx*lely)
     $              , w2p(lx2*ly2*lelx*lely)

      common /outavg/ uavg_z(lx1,ly1,lz1,lelv)
     $              , vavg_z(lx1,ly1,lz1,lelv)
     $              , wavg_z(lx1,ly1,lz1,lelv)
     $              , pavg_z(lx2,ly2,lz2,lelv)
     $              , urms_z(lx1,ly1,lz1,lelv)
     $              , vrms_z(lx1,ly1,lz1,lelv)
     $              , wrms_z(lx1,ly1,lz1,lelv)
     $              , prms_z(lx2,ly2,lz2,lelv)

      common /ctorq/ dragx(0:maxobj),dragpx(0:maxobj),dragvx(0:maxobj)
     $             , dragy(0:maxobj),dragpy(0:maxobj),dragvy(0:maxobj)
     $             , dragz(0:maxobj),dragpz(0:maxobj),dragvz(0:maxobj)
c
     $             , torqx(0:maxobj),torqpx(0:maxobj),torqvx(0:maxobj)
     $             , torqy(0:maxobj),torqpy(0:maxobj),torqvy(0:maxobj)
     $             , torqz(0:maxobj),torqpz(0:maxobj),torqvz(0:maxobj)
c
     $             , dpdx_mean,dpdy_mean,dpdz_mean
     $             , dgtq(3,4)
```

```
      if (ax1.ne.lx1 .or. ay1.ne.ly1 .or. az1.ne.lz1) then
         if(nid.eq.0) write(6,*)
     $     'ABORT: wrong size of ax1,ay1,az1 in avg_all(), check SIZEu!'
         call exitt
      endif
      if (ax2.ne.lx2 .or. ay2.ne.ay2 .or. az2.ne.lz2) then
         if(nid.eq.0) write(6,*)
     $     'ABORT: wrong size of ax2,ay2,az2 in avg_all(), check SIZEu!'
         call exitt
      endif

      ntot  = nx1*ny1*nz1*nelv
      ntott = nx1*ny1*nz1*nelt
      nto2  = nx2*ny2*nz2*nelv


      ! initialization
      if (icalld.eq.0) then
         icalld = icalld + 1
         atime  = 0.
         timel  = time

         call rzero(uavg,ntot)
         call rzero(vavg,ntot)
         call rzero(wavg,ntot)
         call rzero(pavg,nto2)
         do i = 1,ldimt
            call rzero(tavg(1,1,1,1,i),ntott)
            call rzero(tavg(1,1,1,1,i),ntott)
         enddo

         call rzero(urms,ntot)
         call rzero(vrms,ntot)
         call rzero(wrms,ntot)
         call rzero(prms,nto2)
         do i = 1,ldimt
            call rzero(trms(1,1,1,1,i),ntott)
         enddo

         call rzero(vwms,ntot)
         call rzero(wums,ntot)
         call rzero(uvms,ntot)
      endif

      dtime = time  - timel
      atime = atime + dtime

      ! dump freq
      iastep = param(68)
c      if  (iastep.eq.0) iastep=param(15)   ! same as iostep
c      if  (iastep.eq.0) iastep=500

      ifverbose=.false.
      if (istep.le.10) ifverbose=.true.
      if  (mod(istep,iastep).eq.0) ifverbose=.true.

      if (atime.ne.0..and.dtime.ne.0.) then
         if(nid.eq.0) write(6,*) 'Compute statistics ...'
         beta  = dtime/atime
         alpha = 1.-beta

c Compute the average values of the force coeffients
        dragx_avg = alpha*dragx_avg +
     $  beta*(dragx(1)+dragx(2)+dragx(3)+dragx(4))
```

```
         dragy_avg = alpha*dragy_avg +
     $  beta*(dragy(1)+dragy(2)+dragy(3)+dragy(4))

         dragpx_avg = alpha*dragpx_avg + beta*(dragpx(1)
     $      +dragpx(2)+dragpx(3)+dragpx(4))

         dragpy_avg = alpha*dragpy_avg + beta*(dragpy(1)
     $      +dragpy(2)+dragpy(3)+dragpy(4))

         dragvx_avg = alpha*dragvx_avg + beta*(dragvx(1)
     $      +dragvx(2)+dragvx(3)+dragvx(4))
         dragvy_avg = alpha*dragvy_avg + beta*(dragvy(1)
     $      +dragvy(2)+dragvy(3)+dragvy(4))
c
c-----------------------------------------------------------------------
c**********z-average *avg and output*************

       call z_average          (ua,uavg,w1,w2)
       call z_average_transpose(uavg_z,ua)
c
       call z_average          (va,vavg,w1,w2)
       call z_average_transpose(vavg_z,va)
c
       call z_average          (wa,wavg,w1,w2)
       call z_average_transpose(wavg_z,wa)
c
       call z_average          (pa,pavg,w1p,w2p)
       call z_average_transpose(pavg_z,pa)

       call outpost2(uavg_z,vavg_z,wavg_z,pavg_z,tavg,ldimt,'zav')
c-----------------------------------------------------------------------

c     si fanno uscire direttamente gli rms togliendo la media

c
         do i1=1,LX1
            do i2=1,LY1
               do i3=1,LZ1
                  DO i4=1,LELV
                       urms(i1,i2,i3,i4)=urms(i1,i2,i3,i4)-
     &    uavg(i1,i2,i3,i4)*uavg(i1,i2,i3,i4)
                        vrms(i1,i2,i3,i4)=vrms(i1,i2,i3,i4)-
     &    vavg(i1,i2,i3,i4)*vavg(i1,i2,i3,i4)
                          wrms(i1,i2,i3,i4)=wrms(i1,i2,i3,i4)-
     &    wavg(i1,i2,i3,i4)*wavg(i1,i2,i3,i4)
                        enddo
                        enddo
                        enddo
                     enddo
c

         do i11=1,LX2
            do i21=1,LY2
               do i31=1,LZ2
                  DO i41=1,LELV
                       prms(i11,i21,i31,i41)=prms(i11,i21,i31,i41)-
     &    pavg(i11,i21,i31,i41)*pavg(i11,i21,i31,i41)
                        enddo
                        enddo
                        enddo
                     enddo

c-----------------------------------------------------------------------
c**********z-average *avg and output*************
```

```
c       call z_average          (ua1,urms,w1,w2)
c       call z_average_transpose(urms_z,ua1)

c       call z_average          (va1,vrms,w1,w2)
c       call z_average_transpose(vrms_z,va1)

c       call z_average          (wa1,wrms,w1,w2)
c       call z_average_transpose(wrms_z,wa1)

c       call z_average          (pa1,prms,w1p,w2p)
c       call z_average_transpose(prms_z,pa1)

c       call outpost2(urms,vrms,wrms,prms,tavg,ldimt,'rms')
c       call outpost2(urms_z,vrms_z,wrms_z,prms_z,tavg,ldimt,'zrms')
c
c-----------------------------------------------------------------------
c         call outpost(urms,vrms,wrms,prms,trms,'rms')

          do i1=1,LX1
             do i2=1,LY1
                do i3=1,LZ1
                   DO i4=1,LELV
                        urms(i1,i2,i3,i4)=urms(i1,i2,i3,i4)+
     &    uavg(i1,i2,i3,i4)*uavg(i1,i2,i3,i4)
                         vrms(i1,i2,i3,i4)=vrms(i1,i2,i3,i4)+
     &    vavg(i1,i2,i3,i4)*vavg(i1,i2,i3,i4)
                           wrms(i1,i2,i3,i4)=wrms(i1,i2,i3,i4)+
     &    wavg(i1,i2,i3,i4)*wavg(i1,i2,i3,i4)
                         enddo
                         enddo
                         enddo
                      enddo

          do i11=1,LX2
             do i21=1,LY2
                do i31=1,LZ2
                   DO i41=1,LELV
                        prms(i11,i21,i31,i41)=prms(i11,i21,i31,i41)+
     &    pavg(i11,i21,i31,i41)*pavg(i11,i21,i31,i41)
                         enddo
                         enddo
                         enddo
                      enddo

c         call outpost2(uvms,vwms,wums,prms,trms,0    ,'rm2')

          atime = 0.
          time  = time_temp  ! Restore clock

       endif
c
       timel = time
c
       return
       end
```

Figure 30 contains the snapshot of the subroutine Linear_average_s. Basically, this subroutine gives ua1 that contains the average value as the result and has u, w11 and w21 as the entry values. First of all, all the parameters are defined and after filling the parameters by 0s, the average length is calculated by using the command zm1 (calculates the exact coordinate of the desired point) Finally the obtained result is normalized and the linear average is obtained. In order to understand all the commands of NEK5000 it is recommended to look the *Navier5.f* file. Look the url page of [14].

```
      subroutine linear_average_s(ua1,u,w11,w21)
c
c     Compute the linear average of quantity u()
c
      include 'SIZE'
      include 'GEOM'
      include 'PARALLEL'
      include 'WZ'
      include 'ZPER'
c
      real ua1(nx1,nelx,ny1,nely),w11(nx1,nelx,ny1,nely)
      real w21(nx1,nelx,ny1,nely),u(nx1,ny1,nx1,nelv)
      integer e,eg,ex,ey,ez
      integer tmp
c
c     %%%%%%%%%%%%%%%%%%
      nxy = ny1*nely*nx1*nelx
      call rzero(ua1,nxy)
      call rzero(w11,nxy)
      call rzero(w21,nxy)
c     %%%%%%%%%%%%%%%%%%%%
c
      do e=1,nelt     ! loop for all of grid elements
         eg = lglel(e)
         call get_exyz(ex,ey,ez,eg,nelx,nely,nelz)
c
         do k=1,nz1
         do j=1,ny1
         do i=1,nx1

            if (k.eq.1) then
              aa3=(zm1(i,j,k+1,e)-zm1(i,j,k,e))/2
            endif
            if (k.eq.nz1) then
              aa3=(zm1(i,j,k,e)-zm1(i,j,k-1,e))/2
            endif
            if (k.gt.1 .and. k.lt.nz1) then
              aa1=(zm1(i,j,k,e)-zm1(i,j,k-1,e))/2
              aa2=(zm1(i,j,k+1,e)-zm1(i,j,k,e))/2
              aa3=aa1+aa2
            endif
c
            w11(i,ex,j,ey)=w11(i,ex,j,ey)+aa3! length in z direction
            ua1(i,ex,j,ey)=ua1(i,ex,j,ey)+aa3*u(i,j,k,e)  ! sum(l*u)
         enddo
         enddo
         enddo
      enddo
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      call gop(ua1,w21,'+  ',nxy)    ! w21=w21+ua1
      call gop(w11,w21,'+  ',nxy)    ! w21=w21+w11
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      do i=1,nxy
         ua1(i,1,1,1)=ua1(i,1,1,1)/w11(i,1,1,1)   ! Normalize
      enddo
      return
      end
:
```

Figure 30: Subroutine Linear Average Snapshot

List of Figures